A12

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04Q 3/00 — A2

(11) International Publication Number: WO 99/26429

(43) International Publication Date: 27 May 1999 (27.05.99)

(21) International Application Number: PCT/US98/24493

(22) International Filing Date: 17 November 1998 (17.11.98)

(30) Priority Data:
08/971,588 17 November 1997 (17.11.97) US

(71) Applicant: CRAY RESEARCH, INC. [US/US]; 655-A Lone Oak Drive, Eagan, MN 55121 (US).

(72) Inventors: PASSINT, Randal, S.; 90 Grady Drive, Chippewa Falls, WI 54729 (US). THORSON, Greg; 1119 Sweet Water Close, Altoona, WI 54720 (US). GALLES, Michael, B.; 1112 South Springer Road, Los Altos, CA 94024 (US).

(74) Agent: VIKSNINS, Ann, S.; Schwegman, Lundberg, Woessner & Kluth, P.O. Box 2938, Minneapolis, MN 55402 (US).

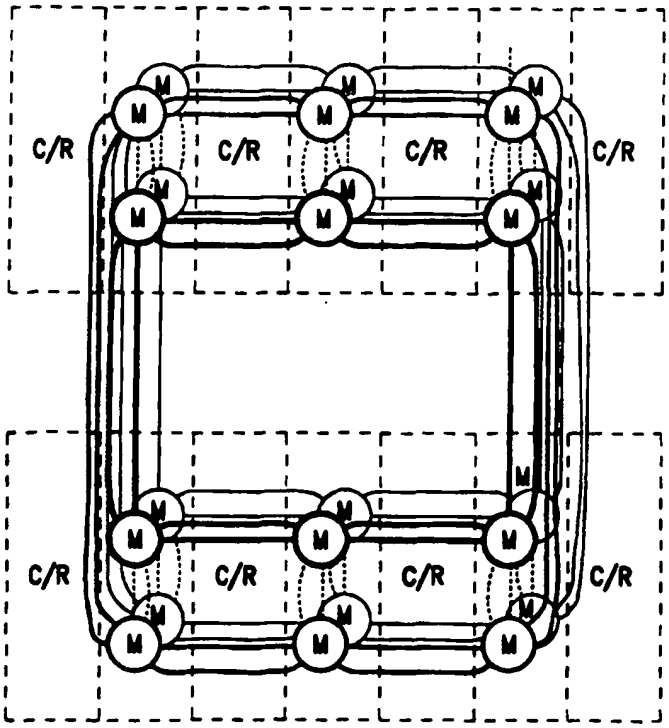
(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: HYBRID HYPERCUBE/TORUS ARCHITECTURE

(57) Abstract

A system and method for communicating between a plurality of processing element nodes within a scalable multiprocessor system. Each processing element node includes at least one processor and memory. A scalable interconnect network includes physical communication links interconnecting the processing element nodes in an n–dimensional topology. Routers in the scalable interconnect network route messages between the plurality of processing element nodes on the physical communication links. The routers are capable of routing messages in hypercube topologies of up to six dimensions, and further capable of routing messages in at least one n–dimensional torus or mesh topology, wherein the topology has at least one of the n dimensions having a radix greater than four.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

HYBRID HYPERCUBE/TORUS ARCHITECTURE

Field of the Invention

The present invention relates generally to the field of high-speed digital data processing systems, and more particularly, to interconnection architectures for interconnecting processor element nodes in multiprocessor computer systems.

Background of the Invention

Multiprocessor computer systems comprise a number of processing element nodes connected together by an interconnect network. Each processing element node includes at least one processing element. The interconnect network transmits packets of information or messages between processing element nodes. Multiprocessor computer systems having up to hundreds or thousands of processing element nodes are typically referred to as massively parallel processing (MPP) systems. In a typical multiprocessor MPP system, every processing element can directly address all of memory, including the memory of another (remote) processing element, without involving the processor at that processing element. Instead of treating processing element-to-remote-memory communications as an I/O operation, reads or writes to another processing element's memory are accomplished in the same manner as reads or writes to the local memory.

In such multiprocessor MPP systems, the infrastructure that supports communications among the various processors greatly affects the performance of the MPP system because of the level of communications required among processors.

Several different topologies have been proposed to interconnect the various processors in such MPP systems, such as rings, stars, meshes, hypercubes, and torus topologies. Regardless of the topology chosen, design goals include a high communication bandwidth, a low inter-node distance, a high network bisection bandwidth and a high degree of fault tolerance.

Inter-node distance is defined as the number of communications links required to connect one node to another node in the network. Topologies are typically specified in terms of the maximum inter-node distance or network diameter: the shortest distance between two nodes that are farthest apart on the network.

Bisection bandwidth is defined as the number of links that would be severed if the network were to be bisected by a plane at a place where the number of links between the two halves is a minimum. In other words, bisection bandwidth is the number of links connecting two halves of the network where the halves are chosen as the two halves connected by the fewest number of links. It is this worst-case bandwidth which can potentially limit system throughput and cause bottlenecks. Therefore, it is a goal of network topologies to maximize bisection bandwidth.

One way to describe bisection bandwidth is in terms of the number of nodes in the network to enable comparison of the relative bisection bandwidth of networks of various sizes. For a network having k nodes, bisection bandwidth is defined in terms of the number of nodes as $x*k$. For example, as described below, a conventional hypercube has a bisection bandwidth of $(\frac{1}{2})k$. This bisection bandwidth remains constant (relative to the number of nodes) at $k/2$ regardless of the dimension of the conventional hypercube.

Note that it may be more appropriate to define bisection bandwidth as the number of communications links times the bandwidth of each link. However, assuming a constant bandwidth/link regardless of the topology, relative bisection bandwidth comparisons among the topologies can simply be addressed in terms of the number of links. Therefore, as a matter of convention, bisection bandwidth is defined in this document in terms of the number of communications links.

In a torus topology, a ring is formed in each dimension where information can transfer from one node, through all of the nodes in the same dimension and back to the original node. An n-dimensional torus, when connected, creates a n-dimensional matrix of processing elements. A bi-

directional n-dimensional torus topology permits travel in both directions of each dimension of the torus. For example, each processing element node in the 3-dimensional torus has communication links in both the + and - directions of the x, y, and z dimensions. Torus networks offer several advantages for network communication, such as increasing the speed of transferring information. Another advantage of the torus network is the ability to avoid bad communication links by sending information the long way around the network. Furthermore, a toroidal interconnect network is also scalable in all n dimensions, and some or all of the dimensions can be scaled by equal or unequal amounts.

In a conventional hypercube network, a plurality of microprocessors are arranged in an n-dimensional cube where the number of nodes k in the network is equal to 2^n. In this network, each node is connected to each other node via a plurality of communications paths. The network diameter, the longest communications path from any one node on the network to any other node, is n-links.

One feature of the conventional hypercube is that the maximum distance between any two nodes (i.e., the diameter) in a hypercube having k nodes is given by $\log_2(k)$. Thus, even as the number of nodes increases, the maximum distance between any two nodes only increases as $\log_2$. As a result, the number of nodes, and hence the number of processors or I/O ports, can be doubled while only requiring a unitary increase in the network diameter between any two nodes. Thus, for each increase in the dimension of the topology (i.e., to expand from an n-dimensional to an $(n + 1)$-dimensional hypercube), an additional edge must be connected to each node. Thus, it is axiomatic that to increase the dimension of a hypercube, each node must have an additional port to support the additional connection. As a result, as the dimension of the hypercube increases the number of ports in each node increases as well.

Another advantage of the conventional hypercube is a high bisection bandwidth. With a conventional hypercube, the bisection bandwidth always remains constant at $k/2$, for a hypercube having k nodes.

Several extensions or variations of the conventional hypercube have been proposed and/or implemented in multiprocessor systems. One such variation, presented in Louri, et al., Scalable optical hypercube-based interconnection network for massively parallel computing, Applied Optics, Vol. 33, No. 32, 10 November 1994, pp 7588-7598, is the multi-mesh hypercube. One disadvantage of the multi-mesh hypercube over the conventional hypercube is a large network diameter. For a multi-mesh hypercube made up of an $l \times m$ array of n-dimensional hypercubes, the maximum distance is $((l - 1) + (m - 1) + n)$. A second disadvantage of the multi-mesh hypercube topology is a low bisection bandwidth relative to the number of nodes in the network. For a symmetrical mesh (where $l = m$), the bisection bandwidth is $k/4$, where k is the number of nodes.

A second variation on the conventional hypercube is presented in Malluhi, et al., The Hierarchical Hypercube: A New Interconnection Topology for Massively Parallel Systems, IEEE Transactions on Parallel and Distributed Systems, vol. 5, No. 1, January 1994, pp. 17-30. According to Malluhi, each of the nodes of an n-dimensional hypercube is itself an n'-dimensional hypercube. This topology has two disadvantages over the conventional hypercube: a lower bisection bandwidth and a greater maximum distance for the same number of nodes.

Consider, for example, a three-dimensional hierarchical hypercube according to Malluhi, where each node is itself a three-dimensional hypercube (i.e., $n = 3$, $n' = 3$). In such a network, there are 64 nodes, and the bisection bandwidth is 4 edges, or $k/16$. Contrast this to the conventional hypercube having 64 nodes with a bisection bandwidth of 32 edges, or $k/2$. Also, Malluhi's hierarchical hypercube has a maximum diameter of $(n + 2n')$ which, for the $n = 3$ and $n' = 3$ network of the current example yields a maximum diameter of nine edges. For a conventional 64-node hypercube, the maximum diameter is $\log_2(64)$ = six edges.

A third variation of the conventional hypercube is presented in Kumar et al., Extended Hypercube: A Interconnection Network of Hypercubes,

IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 1, 1 January 1992, pp. 45-57. According to this extended hypercube topology, network commanders (NC's) are used to connect a plurality of *n*-dimensional hypercubes. There are 2^n nodes at each *n*-dimensional hypercube at the first level, one *n*-dimensional cube of NC's at the second level, and one NC at the third level.

Thus, according to the extended hypercube topology, each node of an *n*-dimensional hypercube at one level of hierarchy is connected to a single communication processor at the next level of hierarchy. A plurality of *n*-dimensional hypercubes, each having an NC are connected via the NC's. One disadvantage of this is that each NC provides a single point of failure for its respective hypercube. If the NC fails, the entire hypercube is severed from the network. Another disadvantage is that apparently additional processing capabilities are needed at the NC nodes. It does not appear that conventional routers can be used to implement the NC. Furthermore, each NC requires a large number of edge connections.

Conventional hypercube topology is a very powerful topology that meets many of the system design criteria. However, when used in large systems, the conventional hypercube has some practical limitations. One such limitation is the degree of fanout required for large numbers of processors. As the degree of the hypercube increases, the fanout required for each node increases. As a result, each node becomes costly and requires larger amounts of silicon to implement.

A hierarchical fat hypercube architecture for parallel processing systems is described in U.S. Patent 5,669,008 entitled "Hierarchical Fat Hypercube Architecture for Parallel Processing Systems," issued on September 16, 1997. In the '008 patent, a scalable hierarchical topology of two or more levels is used. The first level utilizes conventional n-dimensional hypercube topologies for implementing a multiprocessor infrastructure. Each of the n-dimensional hypercubes are interconnected at a second level which can be one of p dimensions, where p does not have to equal n.

The variations on the basic hypercube topology, as noted above, each have their own drawbacks, depending on the size of the network. Some of these topologies suffer from a large network diameter, while others suffer from a low bisection bandwidth. What is needed is a topology that is well suited to applications requiring a large number of processors; is scalable; and provides a high bisection bandwidth, a wide communications bandwidth, and a low network diameter.

Moreover, as systems increase the number of processors, the number of physical connections required to support the hypercube topology increases significantly, resulting in higher system costs and manufacturing complexities. Therefore, it is desired that systems could be scaled to take advantage of more than one type of topology so that smaller systems and larger systems having divergent design goals related to topology architecture could be accommodated in one system design. Such design goals include a desire to optimize system performance while attempting to minimize overall system costs and to minimize manufacturing complexities.

Summary of the Invention

The present invention provides a system and method for communicating between a plurality of processing element nodes within a scalable multiprocessor system. Each processing element node includes at least one processor and memory. A scalable interconnect network includes physical communication links interconnecting the processing element nodes in an n-dimensional topology. Routers in the scalable interconnect network route messages between the plurality of processing element nodes on the physical communication links. The routers are capable of routing messages in hypercube topologies of up to six dimensions, and further capable of routing messages in at least one n dimensional torus or mesh topology, wherein the topology has at least one of the n dimensions having a radix greater than four.

Brief Description of the Drawings

Figure 1 is block diagram of a multiprocessor computer system.

Figure 2 is a block diagram of one embodiment of the interface between a scalable interconnect network and two nodes, each having two processors.

Figure 3 is a block diagram of one embodiment of the interface between a scalable interconnect network and two nodes, each having four processors.

Figure 4 is a model of a two dimensional (2D) hypercube topology multiprocessor system.

Figure 5 is a model of a three dimensional (3D) hypercube topology multiprocessor system.

Figure 6 is a model of a four dimensional (4D) hypercube topology multiprocessor system.

Figure 7 is a model of a five dimensional (5D) hypercube topology multiprocessor system.

Figure 8 is a model of a six dimensional (6D) hypercube topology multiprocessor system.

Figure 9 is a diagram of a 2 x 2 x 2 three dimensional (3D) torus topology.

Figure 10 is a diagram of an example X dimension configuration for one embodiment of a multiprocessor computer system.

Figure 11 is a physical layout diagram for one embodiment of a multiprocessor computer system having 256 nodes.

Figure 12 is a physical layout diagram for one embodiment of a multiprocessor computer system having 128 nodes.

Figure 13 is a physical layout diagram for one embodiment of a multiprocessor computer system having 512 nodes.

Figure 14 is a physical layout diagram for one embodiment of a multiprocessor computer system having 1024 nodes.

Figure 15 is a physical layout diagram for one embodiment of a multiprocessor computer system having 2048 nodes.

Figure 16 is a physical layout diagram for one embodiment of a multiprocessor computer system having 264 nodes.

Figure 17 is a physical layout diagram for one embodiment of a multiprocessor computer system having 544 nodes

Figure 18 is a block diagram of a router chip according to the present invention.

Figure 19A is a diagram of a router table lookup mechanism according to the present invention.

Figure 19B is a diagram of an example entry for a local router table employed in the router lookup mechanism of Figure 19A.

Figure 20 is a diagram of an example set of routes through a network where a set of destination addresses all have the same local destination address, but have different global destination addresses.

Figure 21 is another diagram of an example set of routes through a network where a set of destination addresses all have the same local destination address, but have different global destination addresses.

Figure 22 is a diagram of a mechanism according to the present invention to accomplish the appropriate virtual channel assignment.

Figure 23 is a diagram illustrating the operation of the mechanism of Figure 22.

Figure 24 is a diagram of another embodiment of a basic processor module.

Figure 25 illustrates an eight processor network topology.

Figure 26 illustrates a thirty-two processor network topology.

Figure 27 illustrates a sixteen processor network topology.

Figure 28 illustrates an alternate thirty-two processor network topology.

Figure 29 illustrates a sixty-four processor network topology.

Figure 30 illustrates a sixty-four processor configuration which uses meta-routers to build larger networks.

Figure 31 illustrates a 128 processor network topology based on the configuration of Figure 30.

Figure 32 illustrates a 256 processor network topology based on the configuration of Figure 30.

Figure 33 illustrates a 512 processor network topology based on the configuration of Figure 30.

Figure 34 illustrates a 1024 processor network topology based on the configuration of Figure 30.

Figure 35 illustrates the use of express links to double the bandwidth within a sixty-four processor building block.

Figures 36 and 37 illustrate the use of additional meta-routers to double the global bisection bandwidth of the computer system.

Figure 38 illustrates the use of additional network planes to increase the global bisection bandwidth of the computer system.

Figure 39 illustrates a thirty-two processor network topology having both stub and interstitial I/O nodes.

Description of the Preferred Embodiments

In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural or logical changes may be made without departing from the scope of the present invention. The following detailed description, therefore, is not to be taken in a limiting sense, and the scope of the present invention is defined by the appended claims.

A representative multiprocessor computer system according to the present invention is indicated generally at 20 in Figure 1. As indicated in Figure 1, multiprocessor computer system 20 includes up to n nodes, such as indicated by a first node 22, a second node 24, and an nth node 26. The nodes are interconnected by a scalable interconnect network 28, which permits multiprocessor computer systems 20 to be scale from desk side systems to very large supercomputer configurations.

As illustrated in detail for first node 22, each node in multiprocessor computer system 20 includes at least one processor, such as a first processor 30 and a second processor 32 for node 22. An interface circuit 34 interfaces with scalable interconnect network 28 and communicates with a memory and directory 36 and an input/output crossbar subsystem 38.

Although the multiprocessor computer system 20 illustrated in Figure 1 provides one example environment to implement the below-described hypercube/torus scalable architecture according to the present invention, the present invention is in no way limited to this particular application environment. In fact, many alternative environments using alternative node and interface circuit configurations can be utilized. To a large extent, the topology according to the present invention, as implemented in scalable interconnect network 28, is independent of the complexity of the nodes, such as nodes 22, 24, and 26, interconnected by that topology.

Figure 2 illustrates, in block diagram form, one embodiment of the interface between scalable interconnect network 28 and two nodes 22 and 24. In this embodiment, scalable interconnect network 28 includes router chips, such as indicated at 50. Router chip 50 includes eight ports 52, 54, 56, 58, 60, 62, 64 and 66. Router ports 52 and 54 are respectively coupled to +X dimension physical communication link 70 and -X dimension physical communication link 72. Router ports 56 and 58 are respectively coupled to +Y dimension physical communication link 74 and -Y dimension physical communication link 76. Router ports 60 and 62 are respectively coupled to +Z dimension physical communication link 78 and -Z dimension physical communication link 80. Router port 64 communicates with node 22 and router port 66 communicates with node 24.

As indicated, router port 64 communicates with node 22 via interface chip 34. Similarly, router port 66 communicates with node 24 via interface chip 34'. In node 22, interface chip 34 communicates with processors 30 and 32. In node 24, interface chip 34' communicates with processors 30' and 32'.

Therefore, as illustrated in Figure 2, this implementation of scalable interconnect network 28 transmits packets of information between the processor nodes in the + and - directions of three dimensions and routes packets to two nodes which both include two processors. In other words, one router chip 50 communicates directly with four processors (30, 32, 30' and 32') and six physical communication links (70, 72, 74, 76, 78 and 80).

Figure 3 illustrates, in block diagram form, another embodiment of the interface between a scalable interconnect network 128 and two nodes 122 and 124. In this embodiment, scalable interconnect network 128 includes router chips, such as indicated at 150. Router chip 150 includes eight ports 152, 154, 156, 158, 160, 162, 164 and 166. Router ports 152 and 154 are respectively coupled to +X dimension physical communication link 170 and -X dimension physical communication link 172. Router ports 156 and 158 are respectively coupled to +Y dimension physical communication link 174 and -Y dimension

physical communication link 176. Router ports 160 and 162 are respectively coupled to +Z dimension physical communication link 178 and -Z dimension physical communication link 180. Router port 164 communicates with node 122 and router port 166 communicates with node 124.

As indicated, router port 164 communicates with node 122 via interface chip 134. Similarly, router port 166 communicates with node 124 via interface chip 134'. In node 122, interface chip 134 communicates with processors 130, 131, 132, and 133. In node 124, interface chip 134' communicates with processors 130', 131', 132', and 133'.

Therefore, as illustrated in Figure 3, this implementation of scalable interconnect network 128 transmits packets of information between the processor nodes in the + and - directions of three dimensions and routes packets to two nodes which both include four processors. In other words, one router chip 50 communicates directly with eight processors (130, 131, 132, 133, 130', 131', 132', and 133') and six physical communication links (170, 172, 174, 176, 178 and 180).

As will be better understood by the following discussion, the router chips according to the present invention, such as router chips 50 and 150, can easily scale and accommodate various topologies. In the embodiments illustrated in Figures 2 and 3 the networks are double bristled in that two nodes are connected to a single router 50/150. In other alternative embodiments, additional ports are added to the router chip to permit additional bristling of nodes or the adding of additional dimensions. For example, if two additional ports were added to make a total of ten router ports, + and - directions of a fourth dimension could be added to the interconnect network. Alternatively, the two additional ports could be used to make a quadruple bristled network where four nodes are connected to a single router. In addition, other modification can be made, such as implementing a single bristled network where only one node is connected to a single router. For example in eight-port router chip 50 having a single bristled implementation, there could be the + and - directions for the X, Y and Z dimension for connecting a torus, plus an additional single direction fourth

dimension for connecting a mesh network. In addition, as illustrated in detail below, the eight router ports of router 50 can be used to create up to six-dimensional hypercube topologies.

Example Hypercube Topologies

Multiprocessor computer system 20 employs eight-port router 50 to form up to six-dimensional hypercube systems. The hypercube topology minimizes average hop counts, provides physical link redundancy and maintains linear bisection bandwidth scalability.

An example two dimensional (2D) hypercube topology multiprocessor system is modeled in Figure 4. In Figure 4, four router chips 50 are employed and are numbered 0 through 3. There are two processor ports from each router, such as those labeled PP from router 0, to couple each router to two nodes to create a double bristled topology. Thus, the doubled bristled 2D topology produces an eight node multiprocessor system having 16 processors in a two processor per node system or 32 processors in a four processor per node system. The router links, such as those labeled RL from node 0 to nodes 1 and 3, form the 2D hypercube topology. In addition, extra ports of each router are used to form express links, such as those labeled EL from node 0 to node 2, resulting in a doubling of the bisection bandwidth of the system.

An example three dimensional (3D) hypercube topology multiprocessor system is modeled in Figure 5. In Figure 5, eight router chips 50 are employed and are numbered 0 through 7. There are two processor ports from each router, such as those labeled PP from router 0, to couple each router to two nodes to create a double bristled topology. Thus, the doubled bristled 3D topology produces an 16 node multiprocessor system having 32 processors in a two processor per node system or 64 processors in a four processor per node system. The router links, such as those labeled RL from node 0 to nodes 1, 3, and 7 form the 3D hypercube topology. In addition, extra ports of each router are used to form express links, such as those labeled EL from node 0 to node 5, resulting in a doubling of the bisection bandwidth of the system.

An example four dimensional (4D) hypercube topology multiprocessor system is modeled in Figure 6. In Figure 6, 16 router chips 50 are employed. There are two processor ports from each router, such as those labeled PP from router 0, to couple each router to two nodes to create a double bristled topology. Thus, the doubled bristled 4D topology produces a 32 node multiprocessor system having 64 processors in a two processor per node system or 128 processors in a four processor per node system. The router links, such as those labeled RL from node 0 form the 4D hypercube topology.

An example five dimensional (5D) hypercube topology multiprocessor system is modeled in Figure 7. In Figure 7, 32 router chips 50 are employed. There are two processor ports from each router, such as those labeled PP from router 0, to couple each router to two nodes to create a double bristled topology. Thus, the doubled bristled 5D topology produces a 64 node multiprocessor system having 128 processors in a two processor per node system or 256 processors in a four processor per node system. The router links, such as those labeled RL from node 0 form the 5D hypercube topology. For clarity, only the node 0 links in the fourth and fifth dimensions are shown in Figure 7.

An example six dimensional (6D) hypercube topology multiprocessor system is modeled in Figure 8. In Figure 8, 64 router chips 50 are employed. There are two processor ports from each router, such as those labeled PP from router 0, to couple each router to two nodes to create a double bristled topology. Thus, the doubled bristled 6D topology produces a 128 node multiprocessor system having 256 processors in a two processor per node system or 512 processors in a four processor per node system. The router links, such as those labeled RL from node 0 form the 6D hypercube topology. For clarity, only the node 0 links in the fourth, fifth, and sixth dimensions are shown in Figure 7.

Torus Topologies for Larger System Configurations

Multiprocessor computer system 20 scales from desk size systems to very large super computer configurations. Scalable interconnect network 28 connects multiple nodes via a very high speed, reliable interconnect system. Eight port router 50 can be employed to scale systems beyond 128 nodes in three

dimensional (3D) torus topologies. Non-power-of-two systems can also be configured simply by adding additional cables and reconfiguring routing tables as described below. The below description describes example large scale system configurations and the corresponding system topologies for example systems from 128 to 2048 nodes. Router 50 preferable uses differential signaling to enhance its ability to scale as system sizes grow. As with the multiprocessor topologies described above, two of the eight ports are typically dedicated to connecting from the routers to two separate nodes, such as indicated in Figures 2 and 3. Also as illustrated in Figures 2 and 3, the six remaining ports are employed to form + and - direction connections in three dimensions (X, Y, and Z).

For illustrative purposes, Figure 9 illustrates a 2 x 2 x 2 3D torus topology network in the X, Y, and Z dimensions. Each node in a 3D torus has communication links in both the + and - directions of the X, Y and Z dimensions. A tours topology forms a ring within each dimension where information can transfer from one node, through all the nodes in the same dimension, and back to the origin node.

It is interesting to note that a 4 x 4 x 4 bi-directional 3D torus topology is equivalent to the 6D multiprocessor topology illustrated in Figure 8 having 128 nodes. Thus, as indicated above, the 3D torus topology is utilized for systems having more than 128 nodes, because a double bristled 60 multiprocessor contains 128 nodes.

Global Partitions

As discussed below, each port on router 50 has two router tables (shown in Figures 18 and 19) referred to as a local router table and a global router table. In one embodiment of multiprocessor computer system 20 which is scalable to 2048 nodes, the local router table contains 128 locations and the global router table contains 16 locations. If a packet's source processor is in the same global partition as the destination processor, local tables will describe all of the routes required for the requests to reach their destination and for the response to return to the source. If the destination is in a different global partition, the

global tables are used to describe how to get from one partition to the next. As is described below, because the router tables indicate which output port to take on the next router chip, the router chips which are one hop from the destination global partition also use the local table.

Example X Dimension Configuration

An example X dimension configuration for one embodiment of multiprocessor computer system 20 is illustrated in Figure 10. A router PC board 86 includes four routers, such as router 50, which are labeled R and numbered 0, 1, 2 and 3. In this configuration, the X dimension does not scale as system sizes grow. In this implementation, the X dimension connections are implied in all system topologies greater than 128 nodes. Each of the four routers 50 on router PC board 86 is coupled to two nodes which are labeled N. Each node in the embodiment illustrated in Figure 10 comprises two processors labeled P.

In this embodiment, four routers are connected on the router PC board 86 to form a torus connection of four routers in the X-dimension. The X-dimension does not scale beyond four connections. The four remaining ports of each router chip 50 are connected between router chips to form the Y and Z dimensions for the torus topologies used in larger systems.

Example Torus Topologies

Figure 11 illustrates one embodiment of a system having 256 nodes interconnected with 128 router chips 50 in a double bristled torus topology. In this embodiment there are four X dimension PC router boards 86 in each of eight cabinets 88. Thus, in this embodiment, there are eight nodes coupled to each X dimension router board 86 and 32 nodes within each cabinet 88. The + directions of the Y and Z dimensions are indicated with arrows. As indicated, there are four locations within the X dimensions, eight locations within the Y dimension, and four locations within the Z dimension resulting in a 4 x 8 x 4 torus topology. Also, as illustrated in Figure 11, there is a global partition GP0 and a global partition GP1, where each global partition comprises four cabinets 88.

Figure 12 illustrates one embodiment of a system having 128 nodes interconnected with 64 router chips 50 in a double bristled torus topology. In this embodiment there are four X dimension PC router boards 86 in each of four cabinets 88. Thus, in this embodiment, there are eight nodes coupled to each X dimension router board 86 and 32 nodes within each cabinet 88. The + directions of the Y and Z dimensions are indicated with arrows. As indicated, there are four locations within the X dimensions, four locations within the Y dimension, and four locations within the Z dimension resulting in a 4 x 4 x 4 torus topology. There is only one global partition in the embodiment of Figure 12 having four cabinets 88. This 4 x 4 x 4 torus topology is equivalent to the 6D hypercube topology illustrated in Figure 8.

Figure 13 illustrates one embodiment of a system having 512 nodes interconnected with 256 router chips 50 in a double bristled torus topology. In this embodiment there are four X dimension PC router boards 86 in each of 16 cabinets 88. Thus, in this embodiment, there are eight nodes coupled to each X dimension router board 86 and 32 nodes within each cabinet 88. The + directions of the Y and Z dimensions are indicated with arrows. As indicated, there are four locations within the X dimensions, eight locations within the Y dimension, and eight locations within the Z dimension resulting in a 4 x 8 x 8 torus topology. Also, as illustrated in Figure 13, there are global partitions GP0 through GP3. In this embodiment, global partitions GP0 and GP3 each comprise four cabinets 88, while global partitions GP2 and GP3 are interleaved within a total of eight cabinets.

Figure 14 illustrates one embodiment of a system having 1024 nodes interconnected with 512 router chips 50 in a double bristled torus topology. In this embodiment, there are four X dimension PC router boards 86 in each of 32 cabinets 88. Thus, in this embodiment, there are eight nodes coupled to each X dimension router board 86 and 32 nodes within each cabinet 88. The + directions of the Y and Z dimensions are indicated with arrows. As indicated, there are four locations within the X dimensions, 16 locations within the Y dimension, and eight locations within the Z dimension resulting in a 4 x 16

x 8 torus topology. Also, as illustrated in Figure 14, there are global partitions GP0 through GP7, where each global partition comprises eight half portions of cabinets 88.

Figure 15 illustrates one embodiment of a system having 2048 nodes interconnected with 1024 router chips 50 in a double bristled torus topology. In this embodiment, there are four X dimension PC router boards 86 in each of 32 cabinets 88. Thus, in this embodiment, there are eight nodes coupled to each X dimension router board 86 and 32 nodes within each cabinet 88. The + directions of the Y and Z dimensions are indicated with arrows. As indicated, there are four locations within the X dimensions, 16 locations within the Y dimension, and 16 locations within the Z dimension resulting in a 4 x 16 x 16 torus topology. Also, as illustrated in Figure 15, there are global partitions GP0 through GP15, where each global partition comprises eight half portions of cabinets 88.

Figures 11 through 15 illustrate power-of-two systems. However, as systems scale and size according to the present invention, the systems are not limited to power-of-two configurations. For example, Figure 16 illustrates one embodiment of a 264 node system comprising a first portion 89 which is 4 x 4 x 8 torus within eight cabinets, and a second portion 91 which is 4 x 1 x 1 torus within one cabinet. The 264 node system of Figure 16 comprises global partitions GP0, GP1, and GP3. Global partitions GP0 and GP1 each include four cabinets 88 with 32 nodes within each cabinet. Global partition GP3 includes one cabinet 88, which has 8 nodes.

Another example of a non-power of two system having 544 nodes is illustrated in Figure 17. This 544 node system comprises a first portion 93 which is 4 x 8 x 8 torus within 16 cabinets, and a second portion 95 which is a 4 x 2 x 2 torus within one cabinet. The 544 node system of Figure 17 comprises global partitions GP0 through GP7. Global partitions GP0 and GP1 each include five half portions of cabinets 88, while GP2 through GP7 each include four half portions of cabinets 88.

Other architectures can also be used to advantage. For example, in one embodiment, as is shown in Fig. 24, scalable interconnect network 28 is connected through a module 600 to nodes 602 and 604. In one such embodiment, each module 600 is capable of connecting to up to four planes 601 of network 28 (through ports 603); each plane 601 is capable of configuration as a hierarchical topology with local and global (meta) layers. In the embodiment shown, each module 600 will support up to four parallel planes 601. In one such embodiment, modules 600 and networks 28 are designed so that they are optimized for two such planes 601.

Each node (602, 604) includes an interface chip 606 connected to a memory 608 and two processors 610. In one embodiment each interface chip 606 includes four bidirectional channels 612. Each of the channels 612 can be connected to a separate network plane 601. In one such embodiment, interface chips 606 include bridges 614 used to bridge between two or more network planes 601.

In one embodiment, each node can be connected without the use of routers. One embodiment of a routerless system 620 is shown in Fig. 25. In Fig. 25, two modules 600 are connected via ports 603. The four ports 603 are configured as an upper and a lower ring. Both directions on a ring are used by a given pair of ports 603. (In one embodiment each port 603 supports a transfer of 1.6 GB/s; transfer in both directions therefore provides a bandwidth of 3.2 GB/s).)

In one embodiment, a computer system 20 can be constructed with up to thirty-two processors without the use of routers. Such an embodiment is shown in Fig. 26. Each port 603 is connected to a port 603 of another module 600 as is shown in Fig. 26.

In one embodiment, I/O is provided through an I/O node 618 connected to one or more open router ports as is shown in Fig. 24. In another embodiment, as is shown in Fig. 26 one or more of the links connecting modules 600 pass through an interstitial I/O node 620. Interstitial I/O node 620 is connected between two ports 603 and can receive data from, and transmit data

to, either of the modules 600 connected to it. In addition, in one embodiment, interstitial I/O node 620 includes an internal crossbar switch that can be used as a router in smaller systems. An example of both forms of I/O nodes is shown in Fig. 39.

In another embodiment, systems 20 having 64 or fewer processors are constructed through the use of routers such as router 50. An example of a router-based sixteen processor system 20 is shown in Fig. 27. In the computer system of Fig. 27, a router 50 is connected to one port 603 of each of four modules 600. The two routers 50 and the four modules 600 therefore make up a two-plane system. In one embodiment, routers 50 and modules 600 are enclosed within a single processor subsystem 630. (In the embodiment shown in Fig. 27, two I/O modules stored in a separate cabinet 632 provide I/O for each of the two planes 601.)

Figures 28 and 29 show a thirty-two and sixty-four processor systems 20, respectively, with interstitial I/O.

In one embodiment, systems 20 having more than sixty-four processors 610 are configured as a mesh architecture connecting a plurality of hypercubes. An example of one sixty-four processor group 640 of a larger system 20 is shown in Fig. 30. In the embodiment shown in Fig. 30, systems 20 having more than sixty-four processors contain meta-routers 642 used to bring a plurality of sixty-four processor building blocks 640 together. In one embodiment, meta-routers 642 are formed from the same circuit as the local routers, but are viewed as the second level in the network hierarchy.

In one embodiment, meta-routers 642 are interconnected as meshes. The meshes are 2x1, 2x2, 2x4, 4x4, and 8x4 for 128, 256, 512, 1024, and 2048 processor systems, respectively. Figures 31-34 show these configurations. (A picture showing all of the cabling would be overly complicated, so only the meta-routers and their associated cables are shown. The short thick stubs off of each meta-router indicate the connections to the local routers.)

In the embodiment shown, meta-routers 642 are connected in two-dimensional meshes. The heavy lines in the diagrams highlight one of the eight 2-D meta-meshes. Actually pairs of these four meta-networks may be viewed as 3-D networks with the their third, radix-2, dimension being the dotted links that are shown between meta-routers. (The dotted links are on the router card and, in the case of meta-routers, may be used to avoid deadlock in alternate paths. These dotted links are actually a side effect of the need for dense packaging.) Figure 34 shows a 1024 processor building blocks which can be used to form systems having 2048, 4096, or more processor systems.

As in the examples above, express links can be used to increase the global bisection bandwidth. For example, Fig. 35 illustrates how a small number of cables 650 may be added to double the bandwidth within a sixty-four processor building block such as block 640. In addition, the meta-router count can be doubled as shown in Figs. 36 and 37 to double the global bisection bandwidth at the cost of 1/3 extra network. Finally, as is shown in Fig. 38, system performance can be improved even further through the use of all four network planes 601.

Router Chip

One embodiment of a router chip 50 (or 150) according to the present invention is illustrated in block diagram form in Figure 18. Router chip 50 includes eight differential ports 52, 54, 56, 58, 60, 62, 64, and 66 for coupling to up to eight pairs of unidirectional physical links per router. Four virtual channels, such as indicated at 90, 92, 94, and 96 for port 52, are assigned to each physical channel, where two virtual channels are assigned to requests and two virtual channels are assigned to responses. A more detailed discussion of virtual channels is provided below.

A source synchronous driver/receiver (SSD/SSR) block 98 creates and interprets high-speed, source synchronous signals used for inter-chip communication. A link level protocol (LLP) block 100 interfaces to SSD/SSR block 98 and provides transmission of data between router chips. A router receive block 102 accepts data from LLP block 100, manages virtual channels,

and forwards data to router tables 104 and 106 and a router send block 108. Router receive block 102 includes virtual channel management logic, dynamically allocated memory queues, bypass logic, and fairness logic which ages packets when they fail to make progress. Router send block 108 drives data into LLP block 102 for transmission to other router chips.

Global router table 104 and local router table 106 together form a two level routing table which provides routing information for messages as they pass through the network. Router tables 104 and 106 are indexed by the message destination and direction, and provide a new message direction via an exit port ID. Since routing is pipelined with link arbitration, the routing tables must include instructions as to how to traverse to the next router chip.

For clarity, only port 52 is shown in detail, but all of the eight ports of router chip 50 comprise: virtual channels 90, 92, 94, and 96 a source synchronous driver/receiver (SSD/SSR) block 98; a link level protocol (LLP) block 100; a router receive block 102 ; router tables 104 and 106; and a router send block 108.

A router arbiter block 110 executes two levels of arbitration for the router chip. The first level arbiter performs a wavefront arbitration to selects a near-optimal combination of grants for a given arbitration cycle and informs receiver block 102 which requests won. Ports which are not used during the first level arbitration have a second chance to be granted by the second level or bypass arbiter. Fairness via age comparison is contained within the arbiter block.

A router crossbar block 112 includes a series of multiplexers which control data flow from receiver ports to sender ports. Once arbiter block 110 decides on the winners, arbiter block 110 forwards this information to crossbar block 112 which provides connections from receivers to senders. A router local block 114 is a control point of router chip 50. Router local block 114 provides access to all router controls and status registers including router tables 104 and 106 , error registers (not shown), and protection registers (not shown). Router local block 114 also supports special vector message routing,

which is used during system configuration. In one embodiment, router local block also supports hardware barrier operation.

Message Flow

Messages vary from one to several micropackets in length. Router chip 50 does not assume any particular message length based on header information, but routes a message according to header information until a tail bit is detected. The message header contains all routing and priority information required to complete the message route. Several other fields in the message header are used for memory, processor, and I/O operations. However, only a few fields are decoded by router chip 50. The remaining fields are passed along unchanged as data. Network and node operations are separated as much as possible to permit future networks or future nodes to be interchanged with minimal compatibility problems.

Message header packets follow tail micropackets. Once a micropacket is detected by router chip 50 with its tail bit set in a sideband (discussed below), the next micropacket to the same virtual channel is assumed to be a header. After reset, the first micropacket received by router chip 50 is assumed to be a header. Message body packets are treated as all data, except for sideband information.

A sideband is a field of information that accompanies each micropacket. In one embodiment, router 50 employs the sideband to tag each micropacket with a virtual channel, to communicate virtual channel credits, and to indicate error and tail conditions. Error bit encoding indicates that the micropacket accompanying the error bit indicator encountered a memory ECC error or other type of source error. It is necessary to encode the bit error for every micropacket because, for example, an error might not be detected until the end of a block read and the header of a message will already be routed through the network and cannot indicate an error state.

Message Aging

In one embodiment, each message has an age associated with it and message age influences internal arbitration in router chip 50, where priority is given to older messages. Thus, as a message travels across the network, it ages each time it is stored in a virtual channel buffer. The longer a message waits in a virtual channel buffer, the more it ages. The aging process will continue until the aging limit is reached. In one embodiment, the upper age values are reserved for fixed high priority packets.

Message Routing

In one embodiment, routing chip 50 supports two types of routing which are: 1) table-driven routing for standard, high-speed routing based on internal routing tables; and 2) vector routing for initialization, based on routing instructions included in the message header. Table-driven routing messages are injected into the network with a destination address and an initial direction (i.e., exit port ID), and the routing tables contain the information necessary to deliver the message to its destination. Vector routing requires the source node to completely specify the routing vector when the message is injected into the network.

In one embodiment, nodes 602 and 604 are treated as a single end point from the perspective of the router tables, with the choice of hopping to the neighboring interface chip 606 or routing first on plane 601 being made by the lookup table. In one embodiment, the plane to be used is determined as a function of destination endpoint, low bit of cache line address and a look-up table. In addition, PIOs to a given endpoint travel only on one network plane 601 in order to guarantee ordering.

Vector Routing

The vector routing feature of router chip 50 is used to for access to router registers and some interface circuit registers. Vector routing provides network exploration and routing table initialization and is a low performance routing. The vector routing function permits software to probe the network topology and set up routing tables and ID fields through uncached reads and writes. Once software programs a vector route in a vector route register,

software may execute uncached reads and writes which initiate vector route packets.

In one embodiment, vector route messages are always two micropackets in length. The first micropacket is a standard header, with command encoding indicating read or write, and direction field indicating router core. Whenever a vector route header enters a router chip 50, the vector route header is routed directly to router local block 114. Once inside router local block 114, the second micropacket is examined for the vector route data. Vector route data consists of a vector having vector elements, which each comprise a direction pointer. Direction pointers may include either an output port ID or a vector terminator.

At each hop during the request phase of the vector route, local block 114 examines the current vector and routes according to the right-most vector element. The entire route vector is then shifted right by the number of bits in a vector element, and a return direction port ID is shifted in as the most significant bits. A vector request routed message has reached its destination when the right-most vector element contains binary some indicator, such as for example, 0000 for a four bit vector element.

Once the request packet reaches its destination (the current request vector is all 0s), a response header is formulated and the message is sent back to the port it entered on, but on the reply virtual channel. The new vector is generated such that the least significant nibble becomes the most significant nibble. As a response makes its way through the network, the right-most vector elements are used to route the message and the vector is shifted right. The message eventually reaches the originating node via the same route on which it left.

Table-Driven Routing

All message packets routed by scalable interconnection network 28 during normal operation are routed via routing tables, such as routing tables 104 and 106. Routing tables 104 and 106 are distributed across scalable interconnection network 28 each port of each router and provide a high-speed,

flexible routing strategy which can be adapted through software. Routing tables 104 and 106 determine the path taken by messages between any two nodes in the system. Routing tables 104 and 106 must be programmed in a way which does not introduce any cycles in the directed routing graphs. As dictated by routing tables 104 and 106, the physical message paths are static. The message paths are programmed into routing tables 104 and 106 by software. In the event that a fault develops in scalable interconnection network 28, the static routes dictated by routing tables 104 and 106 can be modified during system operation. However, this software manipulation of routing tables 104 and 106 can be used only for fault avoidance, not for congestion control. Thus, one embodiment scalable interconnection network 28 never routes a message adaptively through the network by "skipping" a desired route when a necessary link is unavailable due to congestion.

Unless noted otherwise, the following routing table discussion refers to a system having two processors per node, such as the system illustrated in Figure 2 above. A router table is indexed by a network destination identification (ID) and provides the next direction a message should travel. Optimally, a router table has a unique entry for each other node in the system. Since this solution is not typically feasible due to the size of the table required, the routing tables, according to the present invention, are broken into a two-level hierarchy. In one embodiment of this two-level hierarchy, there is one router table entry for each of 128 nodes (64 routers) to a given local subnetwork or global partition, as well as a router table entry for each of up to 16 local subnetworks.

Table I at the end of this Description of the Preferred Embodiments section provides one specific example of a TLB mapping showing how the node destination address is broken into global portions for addressing into the global and local tables for a system scalable from 2 to 2048 nodes. The specific example TLB mapping of Table I show how the global and local address bits can be interleaved if desired to accommodate certain physical configurations.

Table II at the end of this Description of the Preferred Embodiments section provides another specific example of a TLB mapping showing how the node destination address is broken into global portions for addressing into the global and local tables for a system scalable from 2 to 256 nodes. The specific example TLB mapping of Table II shows how the global and local address bits are not interleaved for accommodating certain other physical configurations. The specific example TLB mapping of Table II also shows that this system embodiment of up to 256 nodes includes two extra local bits and one extra global to permit future expansion from 272 to 4096 nodes.

In one embodiment, the global destination ID portion must have exactly one bit in the least significant five bits of the destination for systems with greater than 512 nodes. This ensures that an invalidate engine (not shown) maps out nodes appropriately in systems greater than 512 nodes. Without this, the upgrade path would require a 32 node increment to be spread across two half populated cabinets. For systems having less than or equal to 128 nodes, the global bits are preferably avoided all together. This allows for more routing flexibility and less required configuration, as all routing comes from the local table. It should, however, be noted that the scheme used for greater than 128 nodes may also be used for less than or equal to 128 nodes.

A given router chip 50 in the scalable interconnect network 28 must decide whether to use the global or local table to look up a direction to take. The value looked up is not used immediately, but is attached to the packet header and used on the next router chip.

Figure 19A illustrates a router table lookup mechanism illustrated generally at 200. As indicated in Figure 19A, a physical destination ID indicated at 202 includes a global destination ID portion 204 and a local destination ID portion 206. In the embodiment illustrated in Figure 19A, global destination ID 204 includes four bits, while the local destination ID 206 includes seven bits. Global destination ID 204 indexes global routing table 104 having entries 208. Local destination ID 206 indexes local routing table 106 having entries 210. Global destination ID 204 is compared, as indicated at decision point 212, to a

current node's global address stored in register 214. Global destination ID 204 is also compared at decision point 216 to a next -global address stored in register 218. Global destination ID 204 is also compared at decision point 220 to a next +global address stored in register 222.

A direction field from the message header is indicated at 224 and is four bits in the embodiment illustrated Figure 19A. Direction field 224 from the message header 224 is compared at decision point 224 to -global direction bits stored in register 226. Direction field 224 from the message header is also compared at decision point 228 to +global direction bits stored in register 230. The output from decision points 220 and 228 and a +global valid bit stored in register 232 are logically AND'd by an AND gate 234. Similarly, the output of decision points 216 and 224 and a -global valid bit stored in register 236 are logically AND'd by an AND gate 238. The outputs of AND gates 234 and 238, the output of decision point 212, and a force local table bit stored in register 240 are logically OR'd by an OR gate 242.

Figure 19B illustrates an example entry 210 for local router table 106. Local router table entry 210 includes a -global direction field, a +global direction field, and a local direction field. In the embodiment illustrated in Figure 19A, local router table 106 provides the -global direction field on a line 244, the +global direction field on a line 246, and the local direction field on a line 248 to a multiplexer 250. Multiplexer 250 provides one of the direction fields 244, 246, or 248 to one input of a multiplexer 254 based on the state of the outputs of AND gates 234 and 238. Global table 104 provides a global direction field on a line 252 to a second input of multiplexer 254. Multiplexer 254 provides either the global direction field from line 252 or the selected direction field from multiplexer 250 to an output line 256 based on a force local signal provided from OR gate 242. The selected four bit direction field from multiplexer 254 provided on line 256 includes an exit port ID of three bits to determine through which one of the eight ports of the next router chip 50 the message is to be routed, and includes a virtual channel least significant bit (lsb).

In the operation of the embodiment of the router table lookup mechanism of Figure 19A, local router table 106 provides an exhaustive list of local destinations within a local subnetwork, and global router table 104 provides an exhaustive list for each global destination. Local router table 106 is used to determine the next direction when the global bits of the destination ID match the global address of the current router or they match that of the plus or minus neighboring local subnetwork and the output port is the one designated as that which connects directly to that neighboring local subnetwork. When the global bits of the destination ID do not match the global address of the current router chip 50, global router table 104 is used to determine the next direction. In one embodiment, the global ID/port registers 214, 218, 222, 226, 230, 232, and 234 and the force local table register 240 in each router chip 50 are software programmable via a global port definition register (not shown) in router local block 114.

An Example set of routes through a network where a set of destination addresses all have the same local destination address, but have different global destination addresses is illustrated in diagram form in Figure 20. In this example, three routes are performed. All of the three routes only use the local router table entries. The dashed lines represent hops in a given route that carry the local direction field of the local router table; the dotted lines represent hops in a given route that carry the +global direction field in the local router table; and the dotted/dashed lines represent hops in a given route that carry the -global direction field. This particular example does not illustrate any hops that carry the global router table entries.

In Figure 20, the first route passes through an input port 300 of router 12, as indicated by solid line 302. The first route is then routed from router 12 in global partition 2 to router 13 in global partition 3 carrying the +global direction field in the local router table, as indicated by dotted line 304. The first route is then routed from router 13 in global partition 3 to the router 3 (the destination) in global partition 3 carrying the local direction field in the local router table, as indicated by dashed line 306.

In Figure 20, the second route passes through input port 300 of router 12, as indicated by solid line 302. The second route is then routed from router 12 in global partition 2 to the router 2 (the destination) in global partition 2 carrying the local direction field in the local router table, as indicated by dashed line 308.

In Figure 20, the third route passes through input port 300 of router 12, as indicated by solid line 302. The third route is then routed from router 12 in global partition 2 to router 11 in global partition 1 carrying the -global direction field in the local router table, as indicated by dotted/dashed line 310. The third route is then routed from router 11 in global partition 1 to the router 1 (the destination) in global partition 1 carrying the local direction field in the local router table, as indicated by dashed line 312.

Another example set of routes through a network where a set of destination addresses all have the same local destination address, but have different global destination addresses is illustrated in diagram form in Figure 21. In this example, two routes are performed. The dashed lines represent hops in a given route that carry the local direction field of the local router table; the dotted lines represent hops in a given route that carry the +global direction field in the local router table; the single dotted/single dashed lines represent hops in a given route that carry the -global direction field; and the double dotted/single dashed lines represent hops in a given route that carry the global router table entry.

In Figure 21, the first route passes through an input port 400 of router 12, as indicated by solid line 402. The first route is then routed from router 12 in global partition 2 to router 13 in global partition 3 carrying the global entry in the global router table, as indicated by double dotted/single dashed line 404. The first route is then routed from router 13 in global partition 3 to router 14 in global partition 4 carrying the +global direction field in the local router table, as indicated by dotted line 406. The first route is then routed from router 14 in global partition 4 to the router 4 (the destination) in global partition 4 carrying the local direction field in the local router table, as indicated by dashed line 408.

In Figure 21, the second route passes through input port 400 of router 12, as indicated by solid line 402. The second route is then routed from router 12 in global partition 2 to router 11 in global partition 1 carrying the global entry in the global router table, as indicated by double dotted/single dashed line 410. The second route is then routed from router 11 in global partition 1 to router 10 in global partition 0 carrying the -global direction field in the local router table, as indicated by single dotted/single dashed line 412. The second route is then routed from router 10 in global partition 0 to the router 0 (the destination) in global partition 0 carrying the local direction field in the local router table, as indicated by dashed line 414.

Virtual Channels

Four virtual channels, such as indicated at 90, 92, 94, and 96 for port 52, are assigned to each physical channel, where two virtual channels are assigned to requests and two virtual channels are assigned to responses. The scalable interconnect network 28 utilizes the virtual channels to perform deadlock-free routes, create separate request and reply networks, and provide reserved buffers for congestion relief for certain messages. Each message is associated with one of the four virtual channels. In one embodiment, when a message enters a router port, the message is written into the virtual channel buffer indicated by a virtual channel field in the sideband. Congestion control is achieved by allowing messages to switch between the two virtual channels.

In one embodiment, a single physical router link supports interleaving of messages on different virtual channels, but within one virtual channel a message must remain continuous. For example, after router chip 50 sends four micropackets across a physical link on virtual channel 0, router chip 50 can choose to send any number of packets on virtual channels 1-3 before returning to virtual channel 0 to complete the first message. This feature allows router chip 50 to maximize utilization of virtual channel buffers by sending partial messages when only a small amount of space is available at the destination.

In one embodiment, high priority messages can also cut through lower priority messages across a given physical link, assuming the messages do not share the same virtual channel. In order to implement this virtual channel cut-through, each micropacket must be tagged with its virtual channel number in a sideband. This is why virtual channel identification is not contained in the message header in this embodiment.

Deadlock Avoidance

Scalable interconnect network 28 must break cycles of dependencies, as these would cause the network to deadlock. The three types of cyclic dependencies that scalable interconnect network 28 is designed to break are: 1) request-response cycles; 2) turn cycles; and 3) physical cycles. The latter two may only have a semantic difference, but they are differentiated below.

Requests trigger responses which means that cycles are introduced if they use dependent resources. This cycle is broken through the use of virtual channels. All requests travel on one class of virtual channel and all responses on another class of virtual channel. In one embodiment, cache coherency mechanisms in interface circuit guarantee that all traffic fits into one of these two classes, and that no dependencies are introduced from requests to responses or from responses to requests.

As to turn cycles, if turns are not restricted, traffic can loop back on itself. Restrictions to avoid turn cycles are implemented through appropriate loading of the router tables. In one embodiment, the restrictions are as follows: $[z, -y_{nc}, -x_{nc}]$, $[y, -x_{nc}]$, $[x]$. The subscript "nc" indicates "no crossing." That is, do not cross a dateline in that dimension. The concept of datelines is described in detail in the Thorson et al. U.S. Patent 5,659,796, which is assigned to Cray Research, Inc. In other words, the above restrictions cause routing to be performed in three phases as follows:

1. +z, -z, -y, and -x can be routed in any order, but the dateline cannot not be crossed in x or y;
2. +y, -y, and -x can be routed in any order, but the dateline cannot be crossed in x; and

3. x can be completed.

In one embodiment, router chip 50 does not support reflections (i.e., same entrance and exit port). This scheme, combined with the system growth path, provides deadlock avoidance, full use of bisection bandwidth, and multiple paths between nodes.

As to physical cycles, in one embodiment virtual channels are used to break physical cycles in double bristled systems with greater than 128 nodes, such as for a 4 x 8 x 4 double bristled torus topology 256 node system. Physical cycles are only a concern for cycles with greater than four vertices, such as for the Y dimension of the 4 x 8 x 4 torus topology system.

A mechanism according to the present invention to accomplish the appropriate virtual channel assignment is illustrated in diagram form in Figure 22. In this mechanism, the six inter-router ports, such as ports 52, 54, 56, 58, 60, and 62 of router chip 50 illustrated in Figure 2, are divided into three pairs, where each pair defines a dimension. In one embodiment, this pairing is accomplished via a port_mate field 500 stored in a port information register 502 in router chip 50. Port_mate field 500 is a three bit field that matches the lower three bits of the direction field, indicated at 504, on an incoming header packet. The three bit port_mate field 500 is relative to the input port. Table III at the end of this Description of the Preferred Embodiments section provides one embodiment of each input port's relative mate.

The least significant bit (lsb) of the first word of the header micropacket is defined as the "V" bit, indicated at 506, which becomes the lsb of the output virtual channel (nxt_vch_lsb). The next router chip 50 in a given route employs the nxt_vch_lsb to determine which type of virtual channel should be used for routing from the next router chip 50. A virtual channel select enable (vch_sel_en) bit 508 stored in a global information register 510 in router chip 50 is used to control a multiplexer 512. If the vch_sel_en bit 508 is cleared, the nxt_vch_lsb bit is used to indicate the lsb of the next link on line 514 from multiplexer 512. In smaller systems (e.g., less than or equal to 128 nodes in one

embodiment), the vch_sel_en bit 508 is cleared, which causes packets to route from source to destination on the same virtual channel.

On larger systems (e.g., greater than 128 nodes in one embodiment), the vch_sel_en bit 508 is set and the lsb of the virtual channel on line 514 is set using virtual channel lsb on line 256b from the indexed entry from router table 104 or 106 and the port_mate field 500. The indexing of router tables 104 and 106 is described above and illustrated in Figures 19A-B. A comparator 516 compares the direction field on line 256a from the indexed entry from router table 104 or 106 to the port_mate field 500 from port information register 502 to determine whether the output port of the next router chip is the mate to the input port of the next router chip. The output of comparator 516 controls a multiplexer 518, which provides its output to multiplexer 512.

If the vch_sel_en bit 508 is set and the output port is the mate to the input port, the virtual channel lsb on line 256b from lookup table 104 or 106 is logically OR'd with the incoming nxt_vch_lsb bit 506 by an OR gate 520, and this OR'd value is provided as the nxt_vch_lsb on line 514. If the vch_sel_en bit is set and the output port is not the mate to the input port, the nxt_vch_lsb on line 514 is the virtual channel lsb on line 256b from lookup table 104 or 106.

The above mechanism according to the present invention to accomplish the appropriate virtual channel assignment for larger torus systems having at least one radix greater than four can be expressed as follows:

1. Determine if route is straight (i.e, within one dimension) or if route is turning corner (i.e, switching dimensions);
2. If route is turning corner then the nxt_vch_lsb is the virtual channel bit from the routing tables;
3. If route is straight, then the nxt_vch_lsb is obtained by logically ORing the virtual channel bit from routing tables with the virtual channel bit from message header.

The operation of this mechanism for appropriate virtual channel assignment for larger torus systems having at least one radix greater than four is illustrated in diagram form in Figure 23. In Figure 23, only the Y and Z

dimensions are illustrated, because in the embodiment of Figure 10, the X dimension is held at a radix of four. Figure 23 illustrates a route starting on virtual channel 0 from (Y,Z) source (0,0) to (Y,Z) destination (4,1). As illustrated, virtual channel 0 is used to route from router chip (0,0) to router chip (1,0). Virtual channel 0 is used to route from router chip (1,0) to router chip (2,0).

However, in router chip (1,0) the virtual channel bit from the router tables is a one, because of a dateline crossing. The virtual channel bit from the router tables in router chip (1,0) is logically OR's with the incoming V bit, which produces a one. Thus, virtual channel 1 is used to route from router chip (2,0) to router chip (3,0). Once on virtual channel 1, the route stays on virtual channel 1, until a corner is turned. Thus, virtual channel 1 is used to route from router chip (3,0) to router chip (4,0). However, in router chip (3,0), the virtual channel bit from the router tables is a zero, which determines the virtual channel to be used on the corner turn route from router chip (4,0) to router chip (4,1). Thus, virtual channel 0 is used to route from router chip (4,0) to router chip (4,1).

This design approach permits virtual channel switches at a dateline crossing or at all corner turns (i.e., switching dimensions). The lsb of the virtual channel field is used to prevent deadlock in the torus topology in large systems having more than four locations in at least one dimension, such as for a 4 x 8 x 4 torus topology. This allows deadlock free and virtual channel balanced network to be implemented in software. It also allows smaller systems which are deadlock free by nature, to assign certain reference types. For example in one embodiment of a smaller system, I/O references are assigned to virtual channels 1 and 3, while normal processor-to-processor communication are assigned virtual channels 0 and 2.

Resiliency

With the hypercube and torus topologies described above, router chip 50 achieves the goal of always having at least two non-overlapping paths connecting every pair of nodes connected to a router chip 50. Typically, there

are multiple paths available. This permits the system to bypass broken router chips or network links. In addition, in one embodiment, each network link is protected by a check code and link-level logic which retries any corrupted transmissions and provides tolerance of transient errors.

In this way, the hypercube and torus topologies described above can sustain multiple faults and still be reconfigured in a fully connected fashion. In addition, the two-level router table (router tables 104 and 106), when properly implemented, contains only a few models which are connected but cannot be routed, and these typically are unlikely and degenerate cases.

CONCLUSION

The present invention, as described above, permits small systems to be constructed in a hypercube topology which offers high bandwidth connections, low latencies, and physical link redundancy while maintaining linear bisection bandwidth as systems scale. Nevertheless, as systems increase the number of processors, the number of physical connections required to support the hypercube topology increases significantly, resulting in higher system costs and manufacturing complexities. To this end, larger systems according to the present invention can be constructed in a torus topology having more than four locations in at least one dimensions. Such torus topologies according to the present invention permit system performance to be maintained at a relatively high level, yet reduce the number of physical connections between processors. With fewer physical connections between processors, system costs are significantly reduced. Since the torus connection is a ring of processors which grows as systems become larger, multiple torus rings can be formed in the system to keep interconnect bandwidth high and latencies between processors low.

Although specific embodiments have been illustrated and described herein for purposes of description of the preferred embodiment, it will be appreciated by those of ordinary skill in the art that a wide variety of alternate and/or equivalent implementations calculated to achieve the same purposes may be substituted for the specific embodiments shown and described without

departing from the scope of the present invention. Those with skill in the mechanical, electro-mechanical, electrical, and computer arts will readily appreciate that the present invention may be implemented in a very wide variety of embodiments. This application is intended to cover any adaptations or variations of the preferred embodiments discussed herein. Therefore, it is manifestly intended that this invention be limited only by the claims and the equivalents thereof.

Table I: Example TLB Mapping of Dimensions to Physical Node Number

Node Count	Network Dimensions			N_{10}	N_9	N_8	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0
	x	y	z	G_3	L_6	G_2	L_5	G_1	L_4	G_0	L_3	L_2	L_1	L_0
1056 to 2048	4	16	16	z_3	y_3	z_2	y_2	z_1	y_1	z_0	y_0	x_1	x_0	nd
544 to 1024	4	16	8	0	y_3	z_2	y_2	z_1	y_1	z_0	y_0	x_1	x_0	nd
264 to 512	4	8	8	0	0	z_2	y_2	z_1	y_1	z_0	y_0	x_1	x_0	nd
132 to 256	4	8	4	0	0	0	y_2	z_1	y_1	z_0	y_0	x_1	x_0	nd
68 to 128	4	4	4	0	z_1	0	y_1	0	z_0	0	y_0	x_1	x_0	nd
33 to 64	4	4	2	0	0	0	y_1	0	z_0	0	y_0	x_1	x_0	nd
17 to 32	4	2	2	0	0	0	0	0	z_0	0	y_0	x_1	x_0	nd
9 to 16	2	2	2	0	0	0	0	0	z_0	0	y_0	0	x_0	nd
5 to 8	1	1	1	0	0	0	0	0	0	0	y_0	0	x_0	nd
3 to 4	1	1	1	0	0	0	0	0	0	0	0	0	x_0	nd
2	1	1	1	0	0	0	0	0	0	0	0	0	0	nd

Table II: Example TLB Mapping of Dimensions to Physical Node Number

Node Count	Network Dimensions			D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	x	y	z	G_3	L_6	L_5	G_2	G_1	G_0	L_4	L_3	L_2	L_1	L_0
272 to 4096[a]	?	?	?	f_2	f_1	f_0	y_2	y_1	y_0	z_1	z_0	x_1	x_0	nd
132 to 256	4	8	4	0	0	0	y_2	y_1	y_0	z_1	z_0	x_1	x_0	nd
68 to 128	4	4	4	0	0	0	0	y_1	y_0	z_1	z_0	x_1	x_0	nd
33 to 64	4	2	4	0	0	0	0	0	y_0	z_1	z_0	x_1	x_0	nd
17 to 32	4	1	4	0	0	0	0	0	0	z_1	z_0	x_1	x_0	nd
9 to 16	4	1	2	0	0	0	0	0	0	0	z_0	x_1	x_0	nd
5 to 8	4	1	1	0	0	0	0	0	0	0	0	x_1	x_0	nd
3 to 4	1	1	1	0	0	0	0	0	0	0	0	0	x_0	nd
2	1	1	1	0	0	0	0	0	0	0	0	0	0	nd

a. Future expansion

Table III: Encoding of Port_Mate Field

Output Port	PTA Input Port	PTB Input Port	PTC Input Port	PTD Input Port	PTE Input Port	PTF Input Port	PTG Input Port	PTH Input Port
A	n/a	7	6	5	4	3	2	1
B	1	n/a	7	6	5	4	3	2
C	2	1	n/a	7	6	5	4	3
D	3	2	1	n/a	7	6	5	4
E	4	3	2	1	n/a	7	6	5
F	5	4	3	2	1	n/a	7	6
G	6	5	4	3	2	1	n/a	7
H	7	6	5	4	3	2	1	n/a

WHAT IS CLAIMED IS:

1. A scalable multiprocessor system comprising:

a plurality of processing element nodes, each processing element node having at least one processor and memory; and

a scalable interconnect network including:

a first level of interconnect connecting the plurality of processing element nodes in a set of n-dimensional hypercubes, wherein n is greater than one;

a second level of interconnect connecting the hypercubes in a mesh topology;

first routers for routing messages between the plurality of processing element nodes within each hypercube; and

second routers, connected to two or more first routers, for routing messages between first routers connected to different hypercubes.

2. The scalable multiprocessor system of claim 1 wherein the second level of interconnect includes interconnect necessary to form an m-dimensional torus topology, wherein m is greater than two.

3. The scalable multiprocessor system of claim 1 wherein each processing element node includes two processors.

4. The scalable multiprocessor system of claim 1 wherein each processing element node includes four processors.

5. The scalable multiprocessor system of claim 1 wherein each second router includes at least eight router ports.

6. The scalable multiprocessor system of claim 1 wherein each first router includes at least eight router ports.

7. The scalable multiprocessor system of claim 6 wherein two of the at least eight router ports are coupled directly to two corresponding processor element nodes.

8. The scalable multiprocessor system of claim 6 wherein six of the at least eight router ports are coupled directly to the first level of interconnect.

9. The scalable multiprocessor system of claim 1 wherein each router includes router ports and lookup tables at each router port for providing a port direction for exiting from the next router in a given route in the system.

10. The scalable multiprocessor system of claim 9 wherein the lookup tables at each port include a local router table having directions for routing between processor element nodes within a local subnetwork and a global router table having directions for routing between processor element nodes in two subnetworks.

11. The scalable multiprocessor system of claim 1 wherein each first router is coupled to at least two processing element nodes.

12. A method of communicating between processors in a scalable multiprocessor system, comprising:

providing a plurality of hypercubes;

assigning each of a plurality of processors to one of the plurality of hypercubes;

connecting each of the hypercubes within a two-dimensional mesh interconnect;

routing messages between processors assigned to a same hypercube within the hypercube; and

routing messages between processors assigned to different hypercubes across the mesh interconnect.

13. The method of communicating according to claim 12, wherein connecting each of the hypercubes within a two-dimensional mesh interconnect includes:

providing a plurality of meta-routers; and

connecting a router from each of the hypercubes to one of the meta-routers.

14. A method of connecting processors in a scalable multiprocessor system, comprising:

providing a plurality of hypercubes;

assigning each of a plurality of processors to one of the plurality of hypercubes; and

connecting each of the hypercubes within a two-dimensional mesh interconnect, wherein connecting each of the hypercubes within a two-dimensional mesh interconnect includes:

providing a plurality of meta-routers; and

connecting a router from each of the hypercubes to one of the meta-routers.

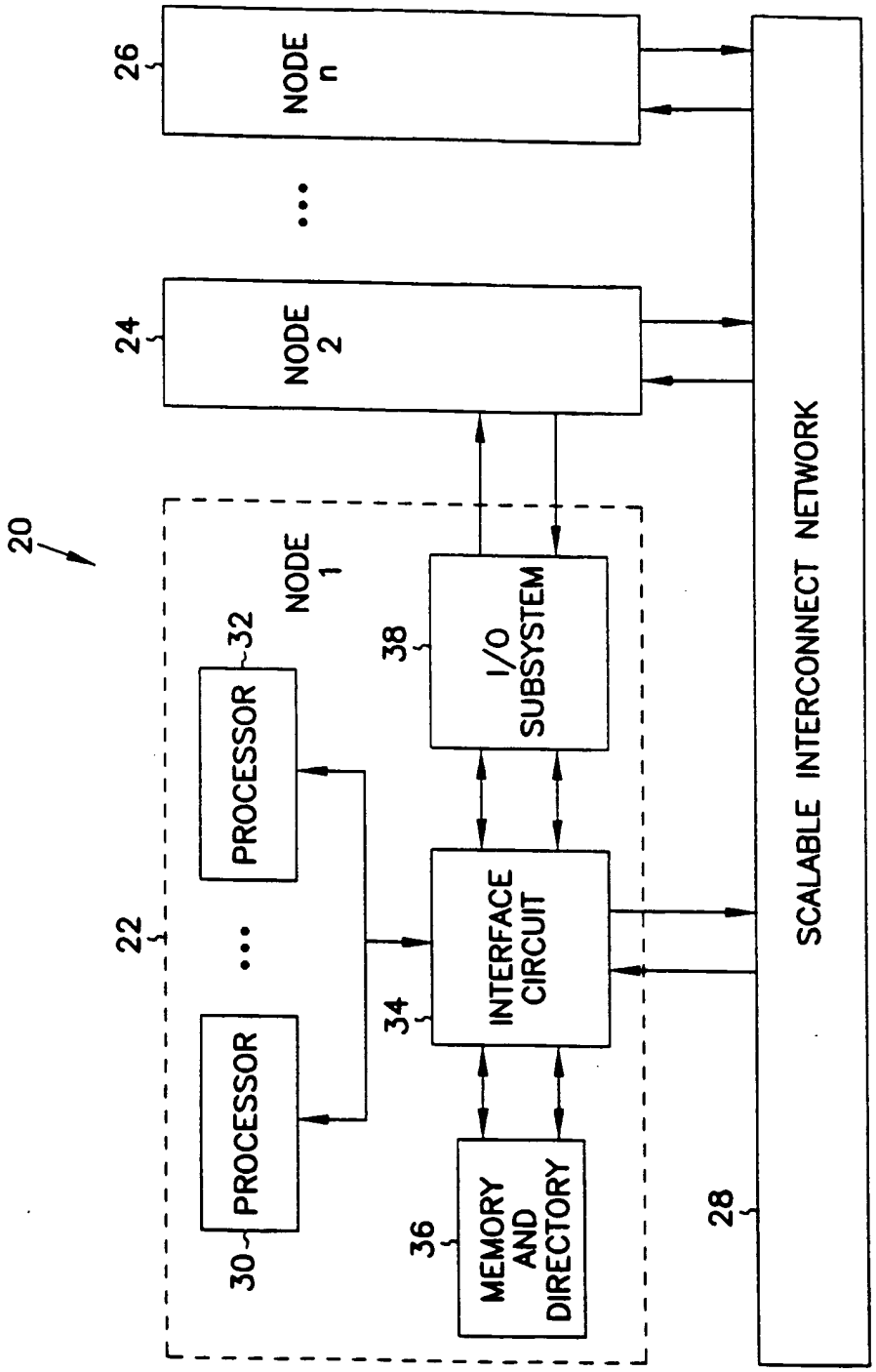

20
NODE 1
22
PROCESSOR
32
PROCESSOR
30
INTERFACE CIRCUIT
34
I/O SUBSYSTEM
38
MEMORY AND DIRECTORY
36
NODE 2
24
NODE n
26
SCALABLE INTERCONNECT NETWORK
28

FIG. 1

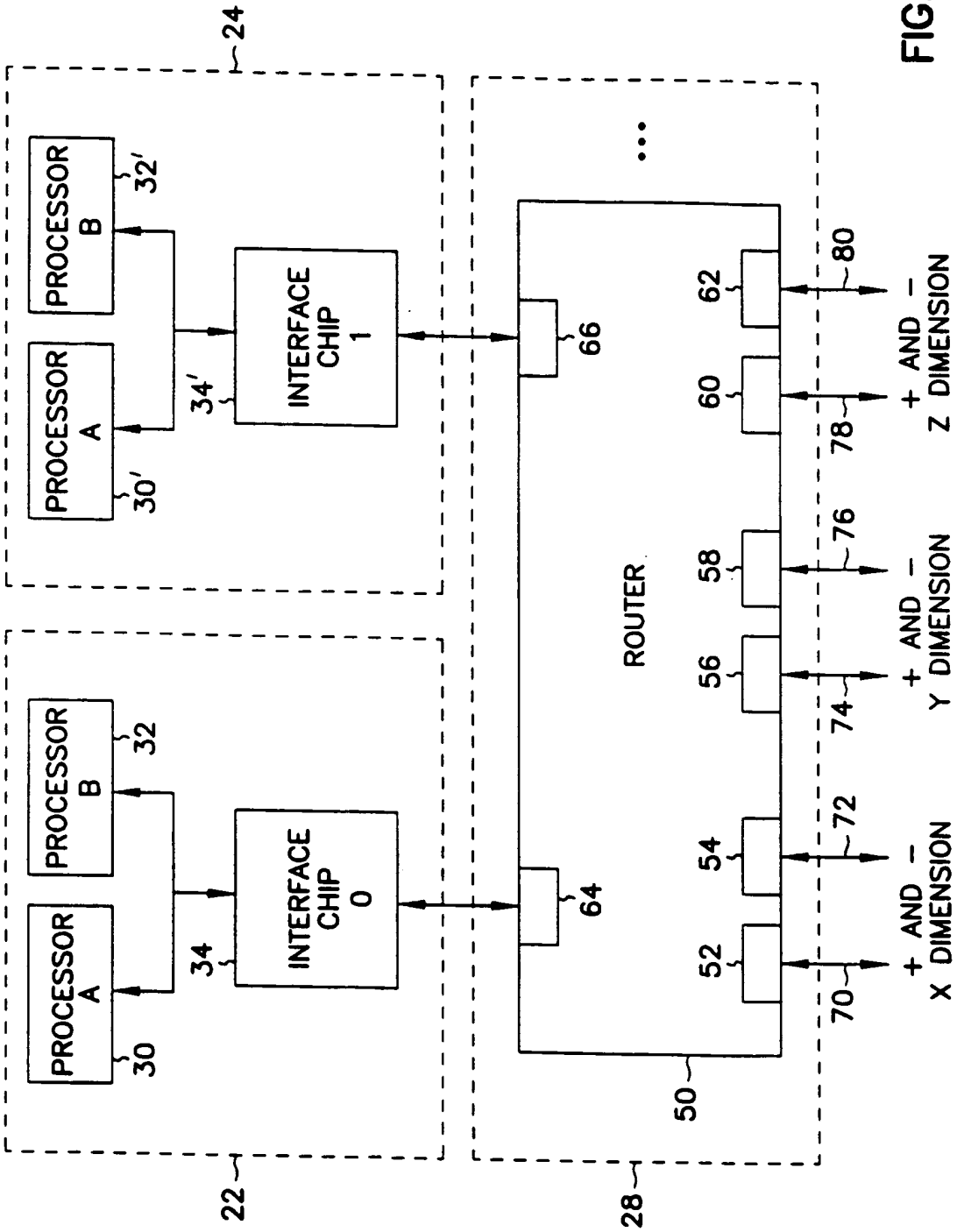

24
PROCESSOR B
32'
PROCESSOR A
30'
34'
INTERFACE CHIP 1
66
PROCESSOR B
32
PROCESSOR A
30
34
INTERFACE CHIP 0
64
ROUTER
50
22
28
62
60
58
56
54
52
80
78
76
74
72
70
+ AND - Z DIMENSION
+ AND - Y DIMENSION
+ AND - X DIMENSION

FIG. 2

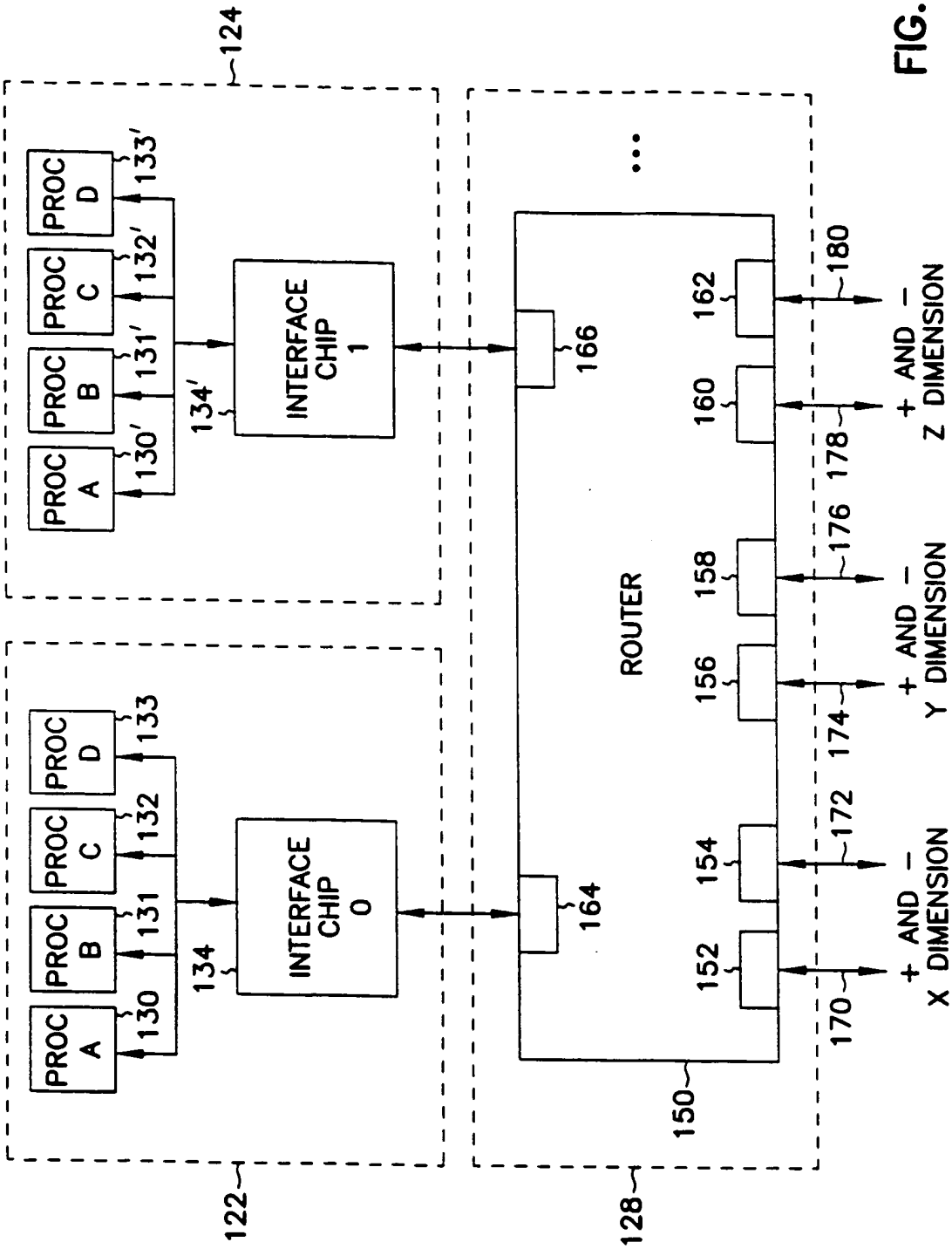

124
PROC A
PROC B
PROC C
PROC D
130'
131'
132'
133'
134'
INTERFACE CHIP 1
166
ROUTER
150
164
INTERFACE CHIP 0
134
130
131
132
133
122
128
152
154
156
158
160
162
170
172
174
176
178
180
+ AND - X DIMENSION
+ AND - Y DIMENSION
+ AND - Z DIMENSION

FIG. 3

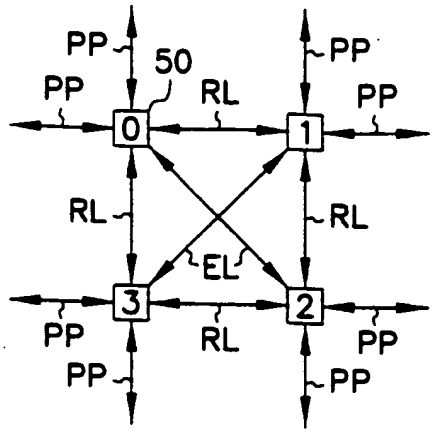

PP
50
PP
PP
RL
PP
0
1
RL
RL
EL
3
2
PP
RL
PP
PP
PP

FIG. 4

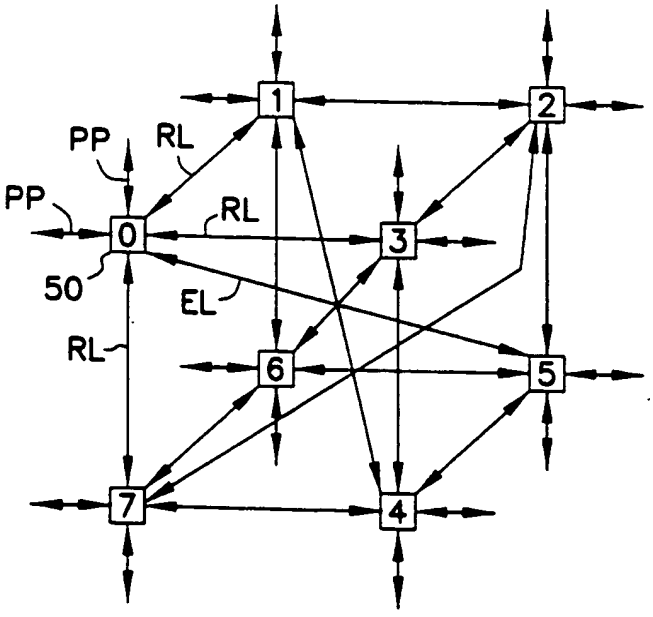

1
2
PP
RL
PP
RL
0
3
50
EL
RL
6
5
7
4

FIG. 5

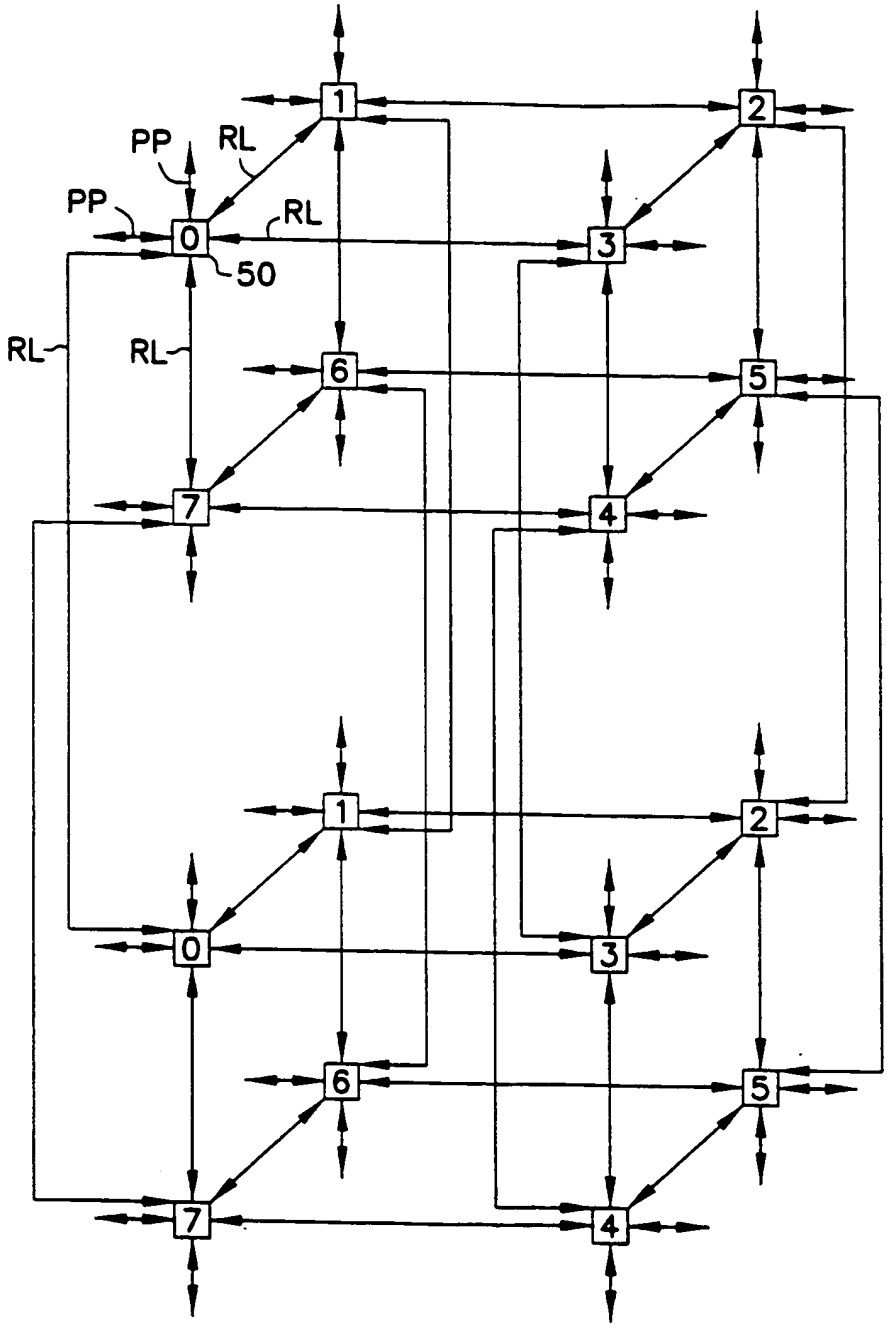


FIG. 6

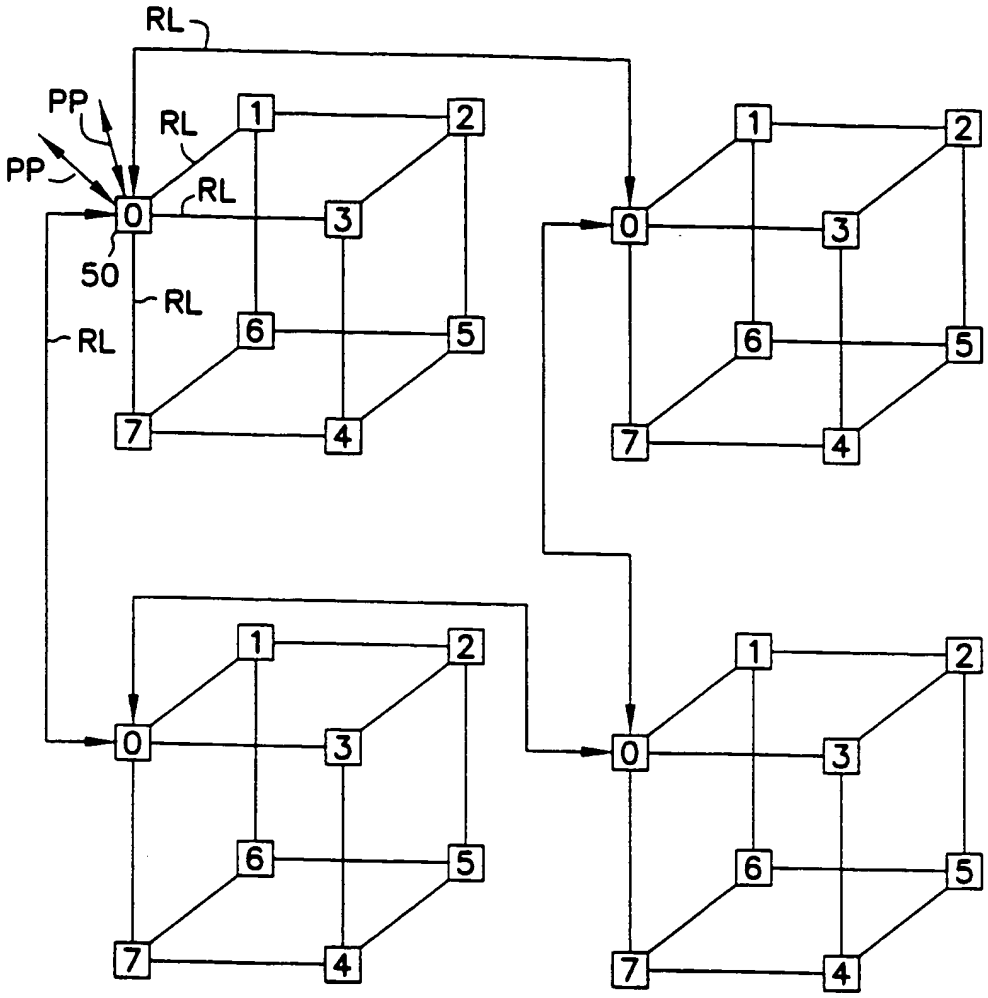


FIG. 7

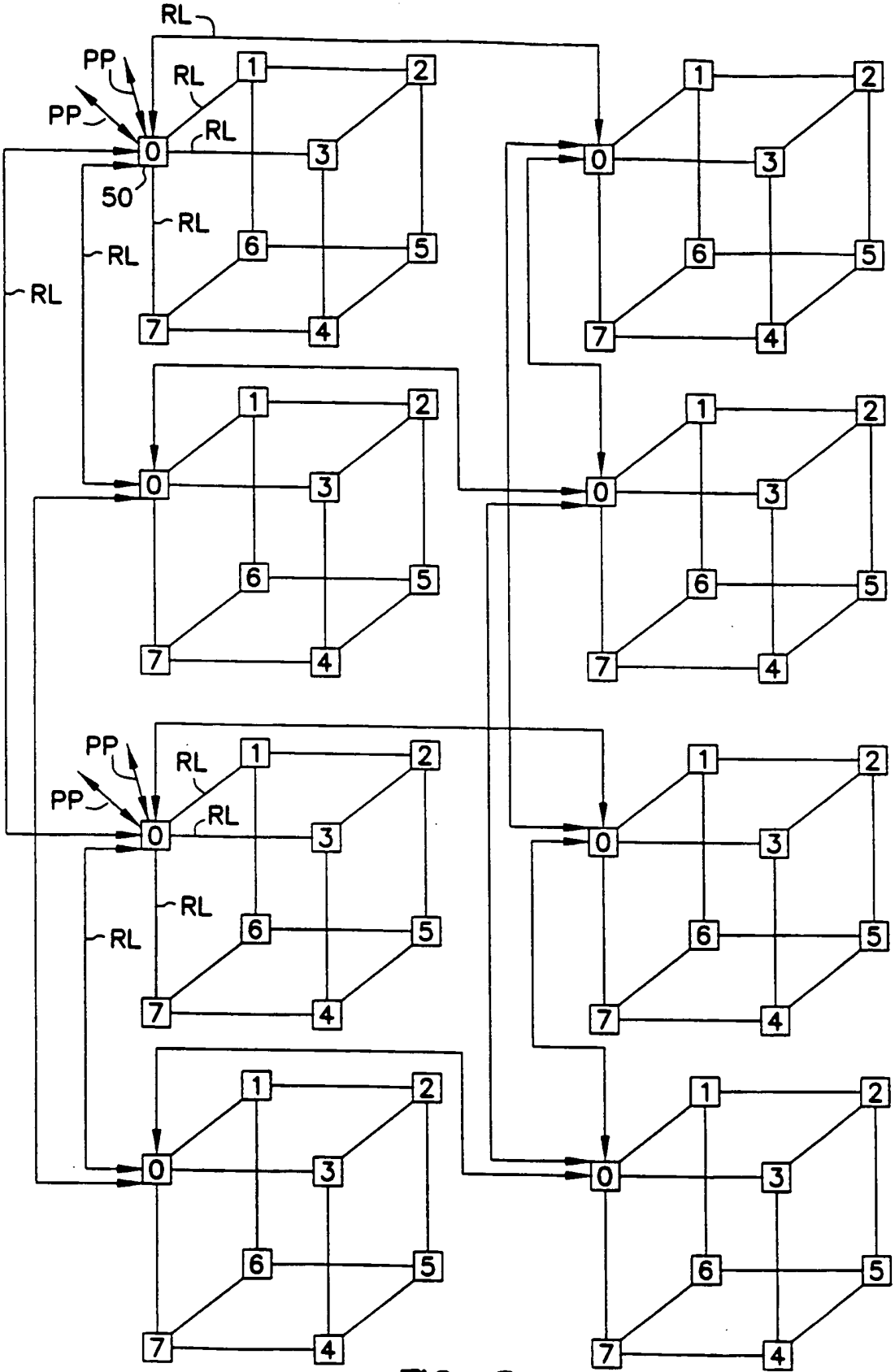
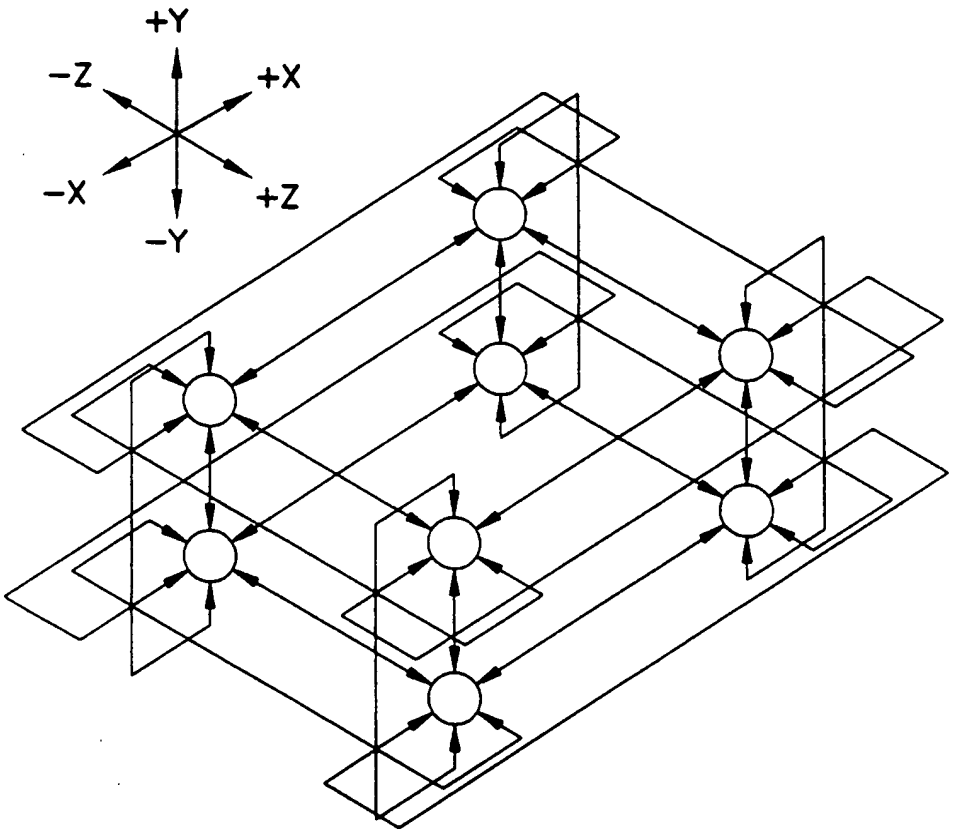


FIG. 8

FIG. 9



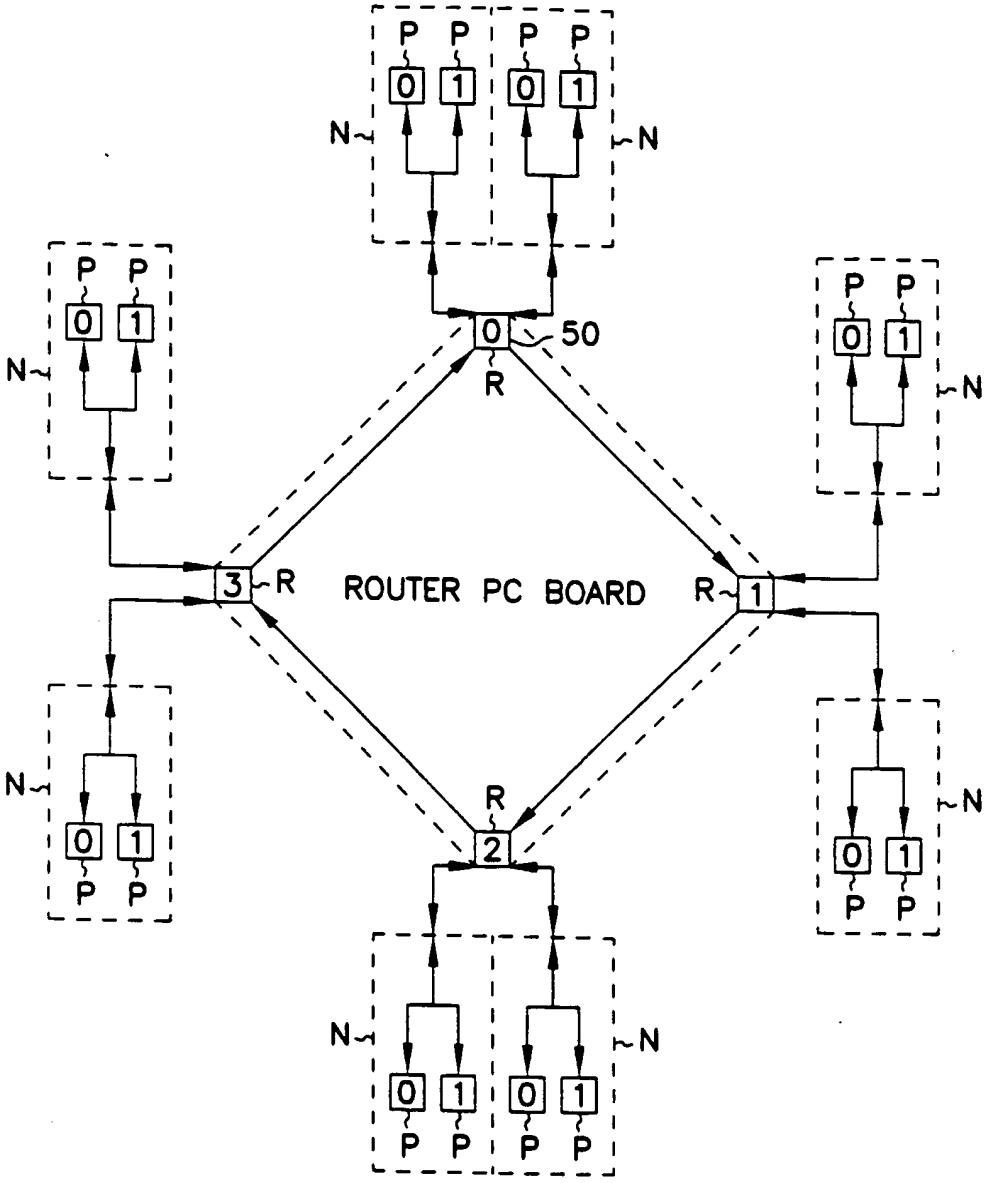

P P P P
0 1 0 1
N N
0
50
R
P P
0 1
N
P P
0 1
N
3 R
ROUTER PC BOARD
R 1
R
2
N
0 1
P P
N
0 1
P P
P P P P
0 1 0 1
N N

FIG. 10

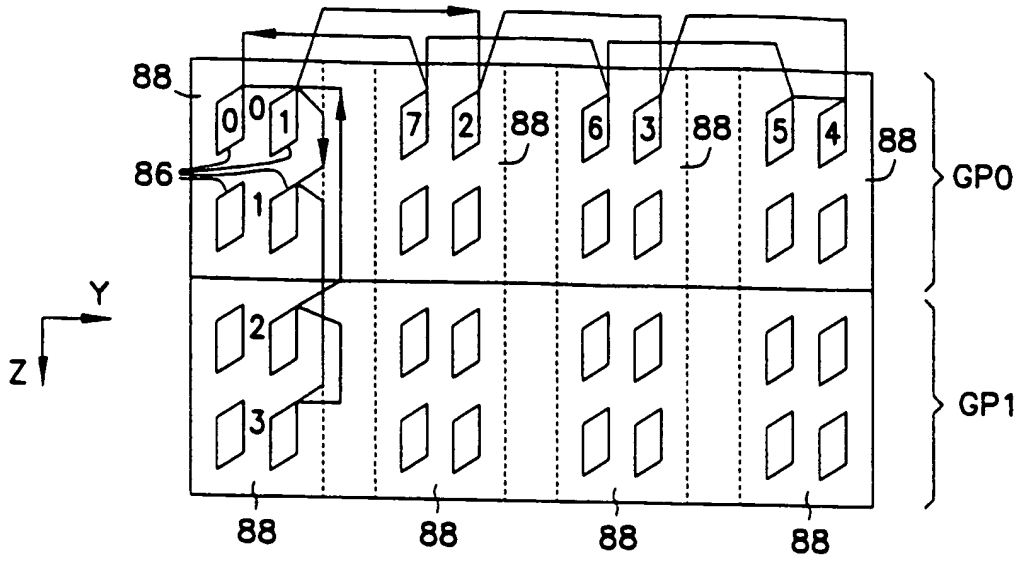

88
0
0
1
7
2
88
6
3
88
5
4
88
86
1
GP0
Y
Z
2
3
GP1
88
88
88
88

FIG. 11

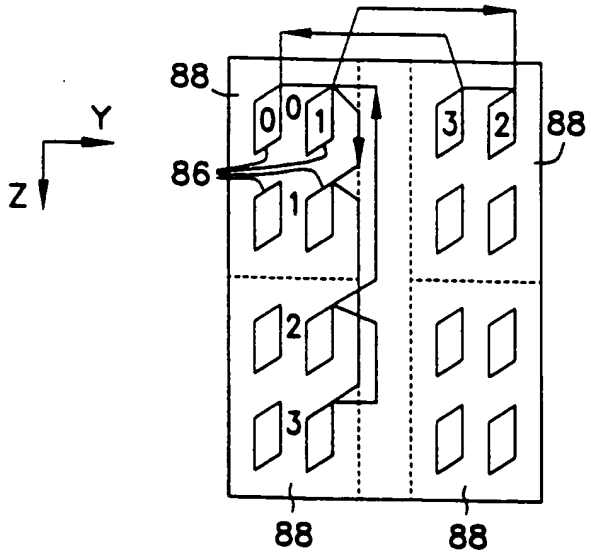

88
0
0
1
3
2
88
86
1
Y
Z
2
3
88
88

FIG. 12

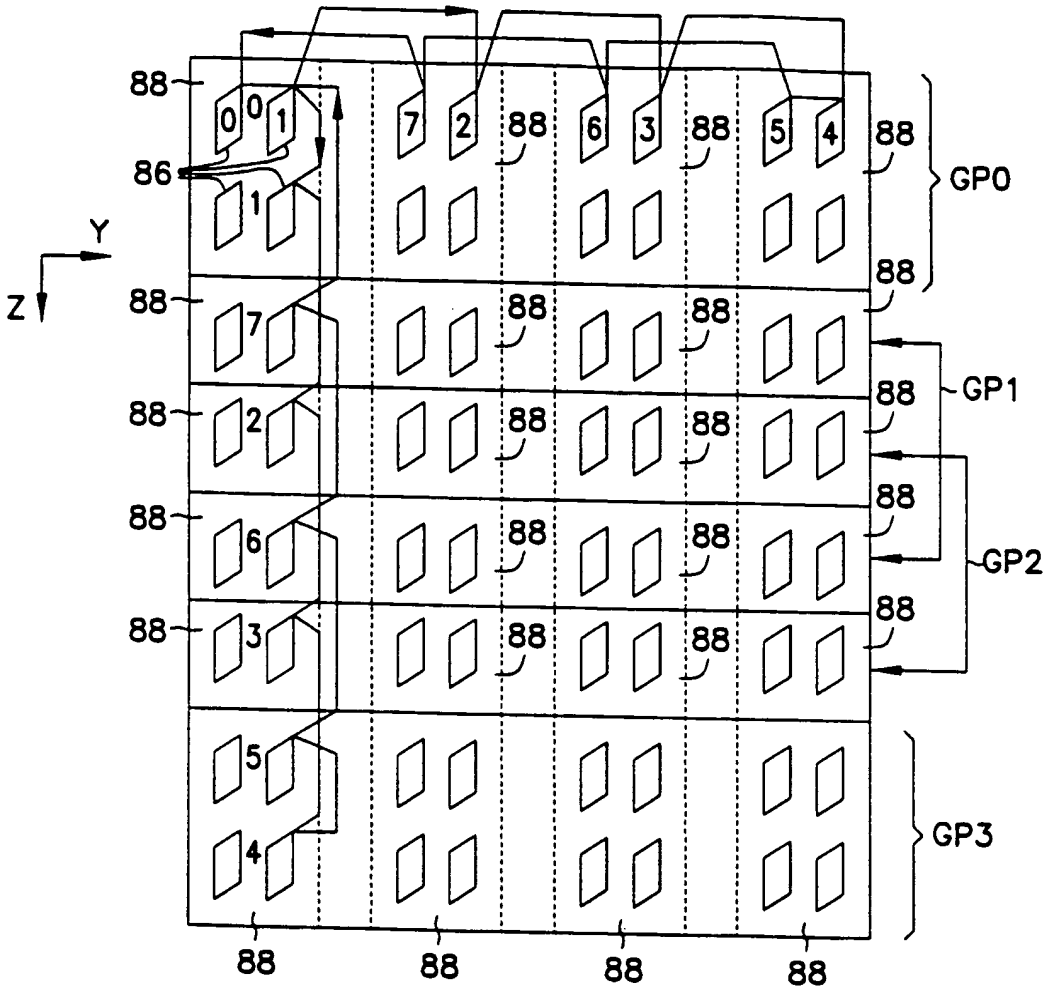

88
0
0
1
7
2
88
6
3
88
5
4
88
GP0
86
1
Y
Z
88
7
88
88
88
GP1
88
2
88
88
88
88
6
88
88
88
88
3
88
88
88
GP2
88
5
4
GP3
88
88
88
88

FIG. 13

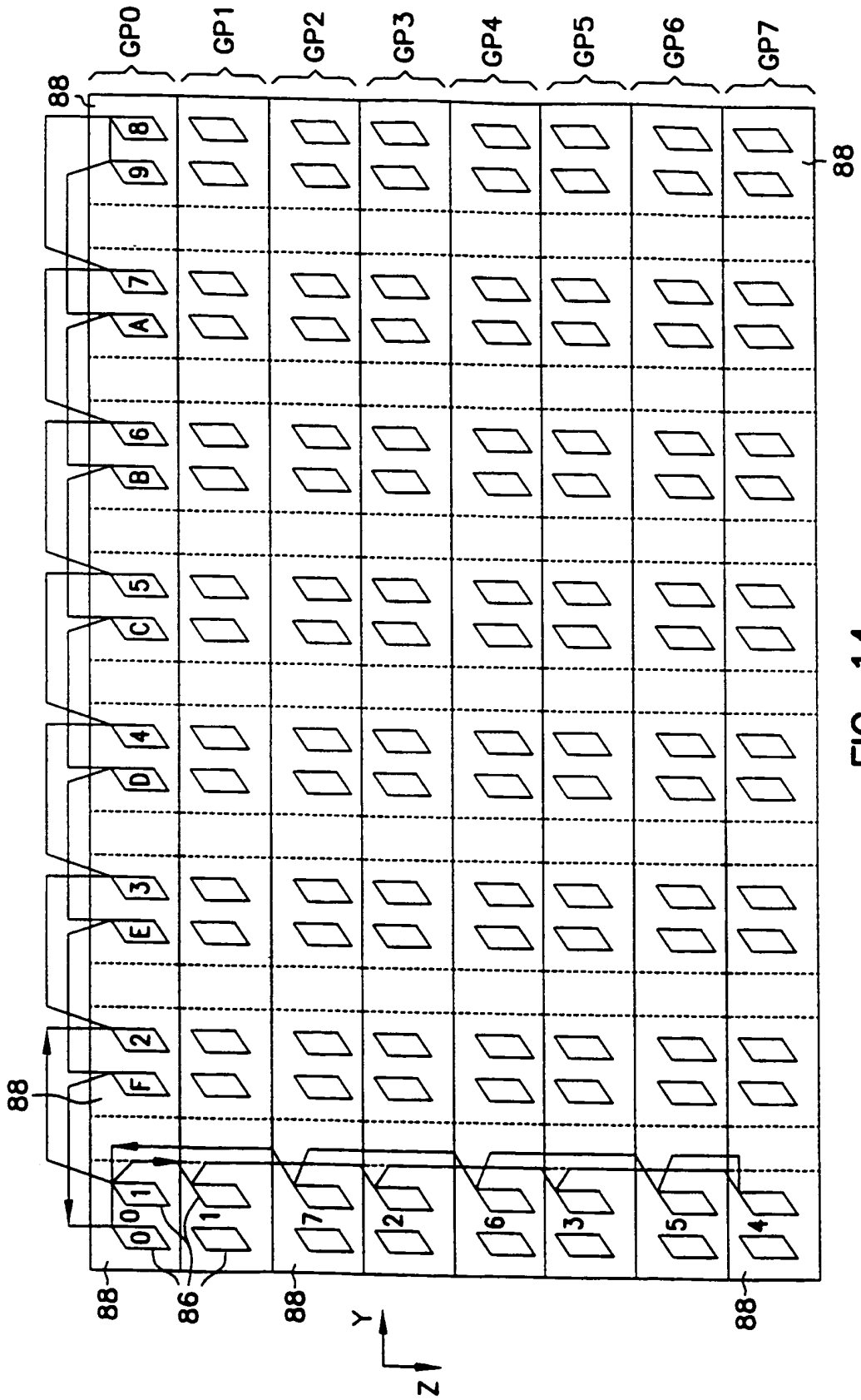


FIG. 14

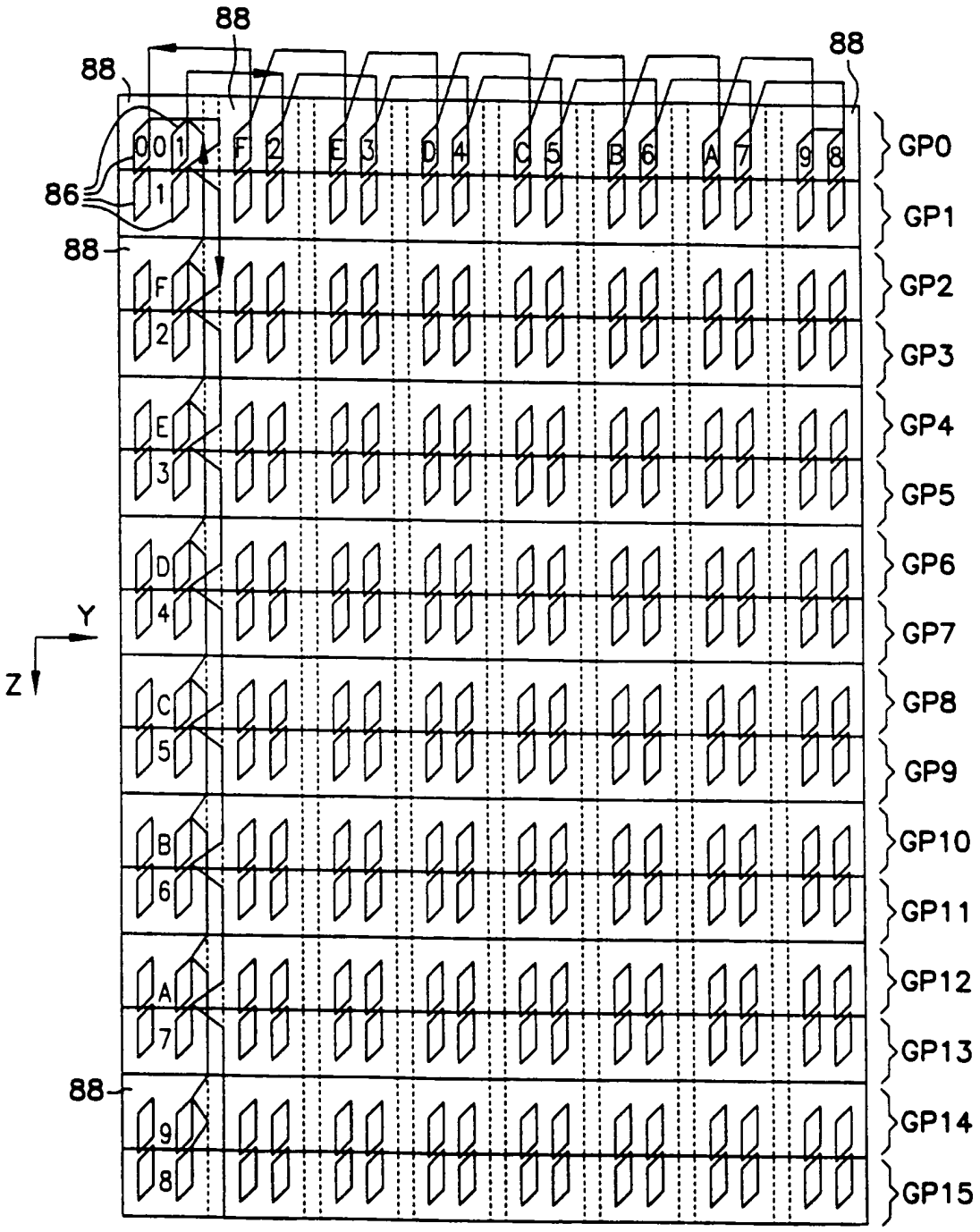

88
88
88
88
86
0
0
1
F
2
E
3
D
4
C
5
B
6
A
7
9
8
1
F
2
E
3
D
4
C
5
B
6
A
7
9
8
GP0
GP1
GP2
GP3
GP4
GP5
GP6
GP7
GP8
GP9
GP10
GP11
GP12
GP13
GP14
GP15
Y
Z

FIG. 15

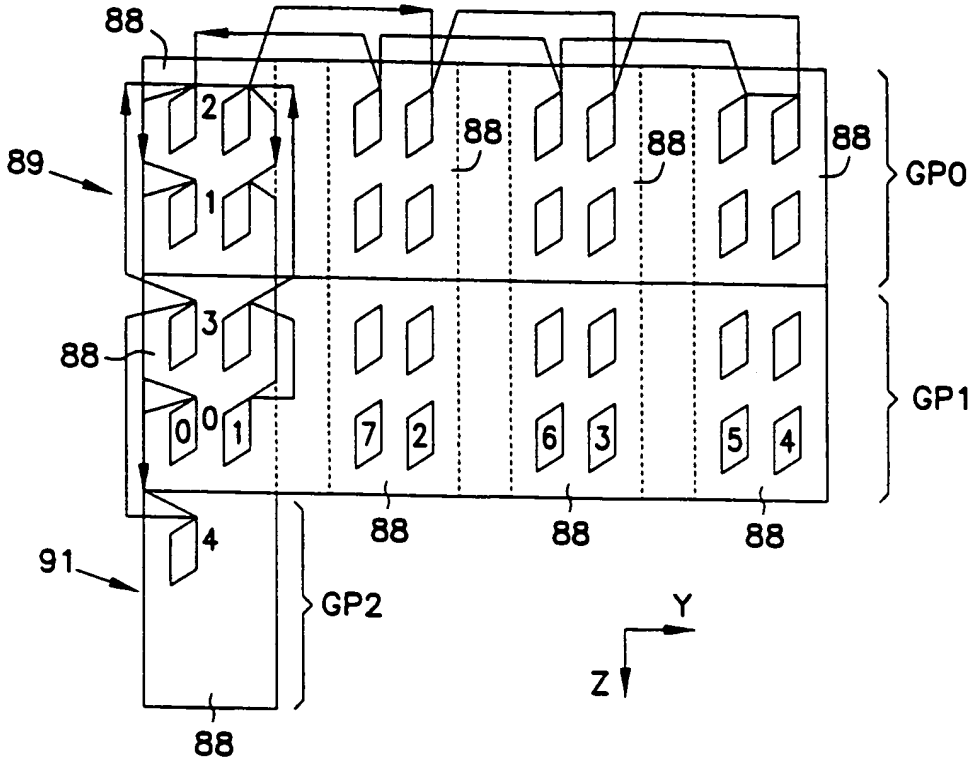


FIG. 16

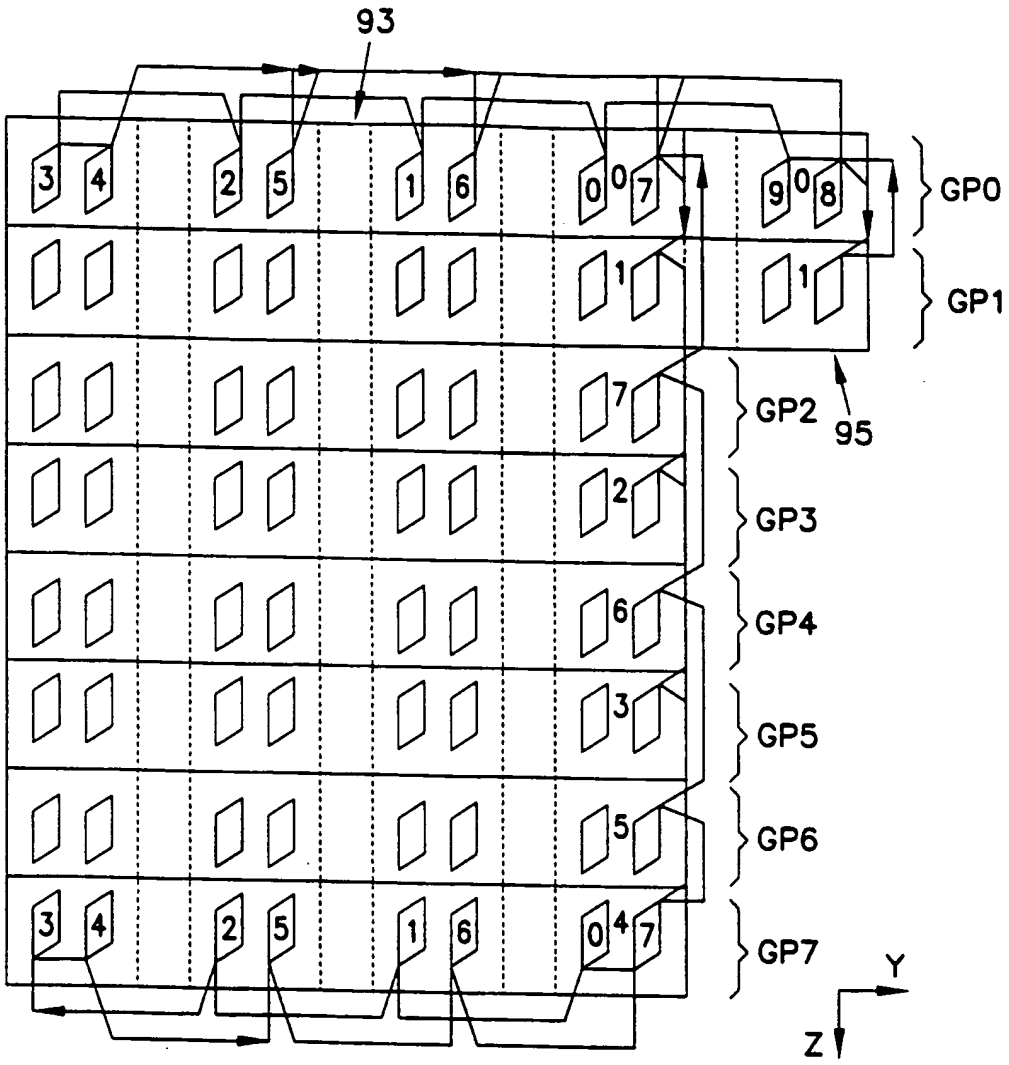


FIG. 17

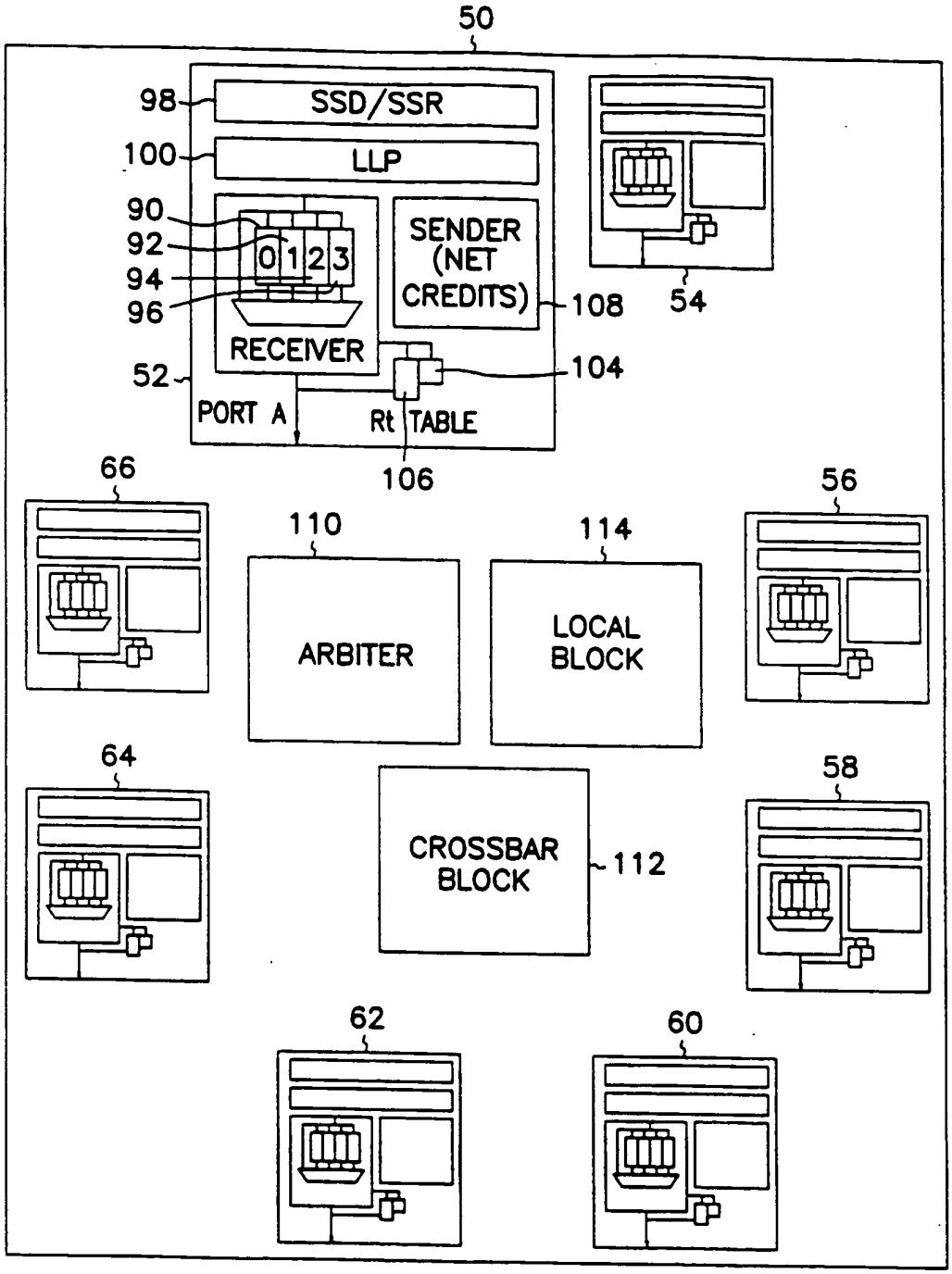

50
98
SSD/SSR
100
LLP
90
92
94
96
0 1 2 3
RECEIVER
SENDER (NET CREDITS)
108
54
52
104
PORT A
Rt TABLE
106
66
110
ARBITER
114
LOCAL BLOCK
56
64
CROSSBAR BLOCK
112
58
62
60

FIG. 18

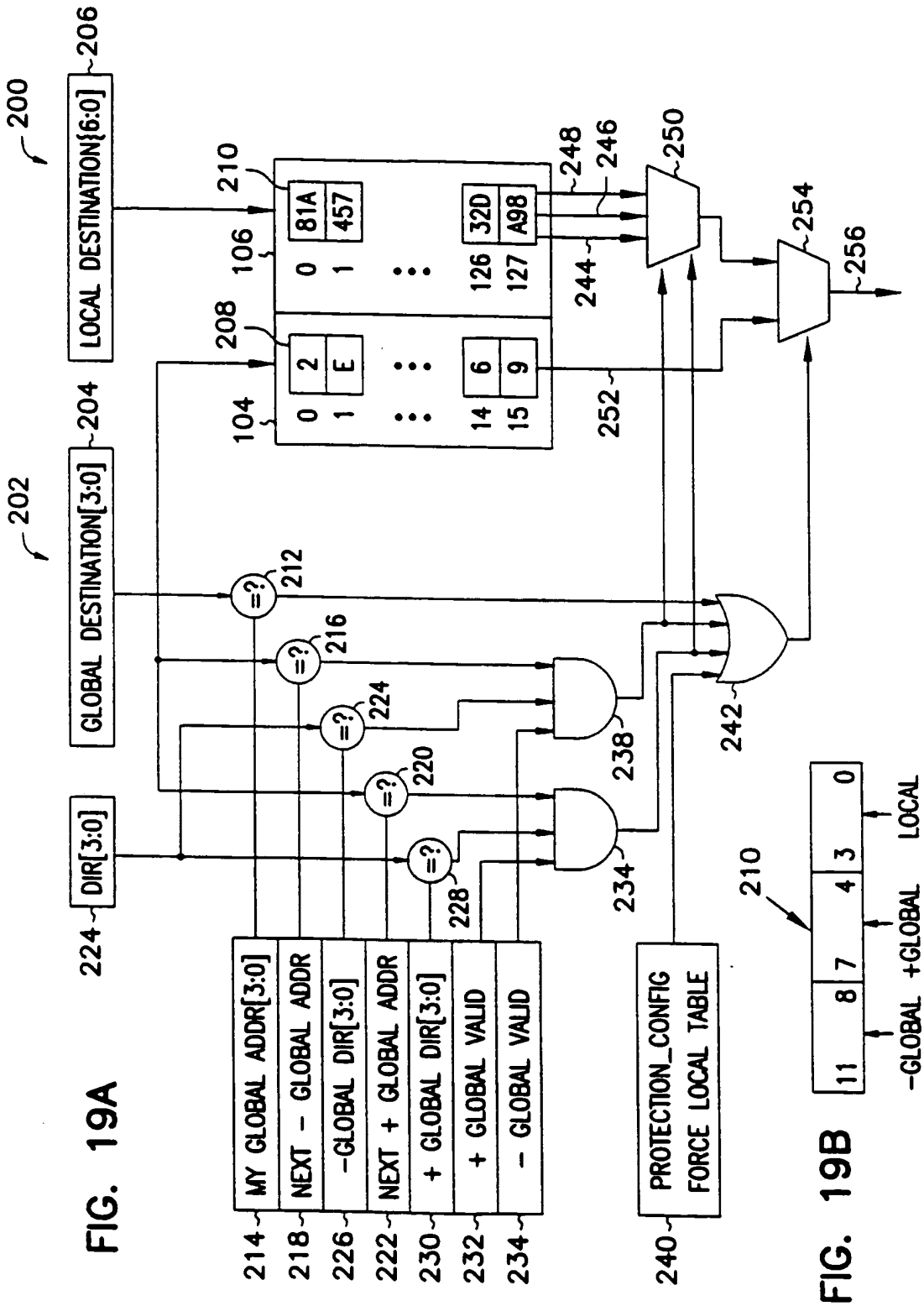


FIG. 19A

FIG. 19B

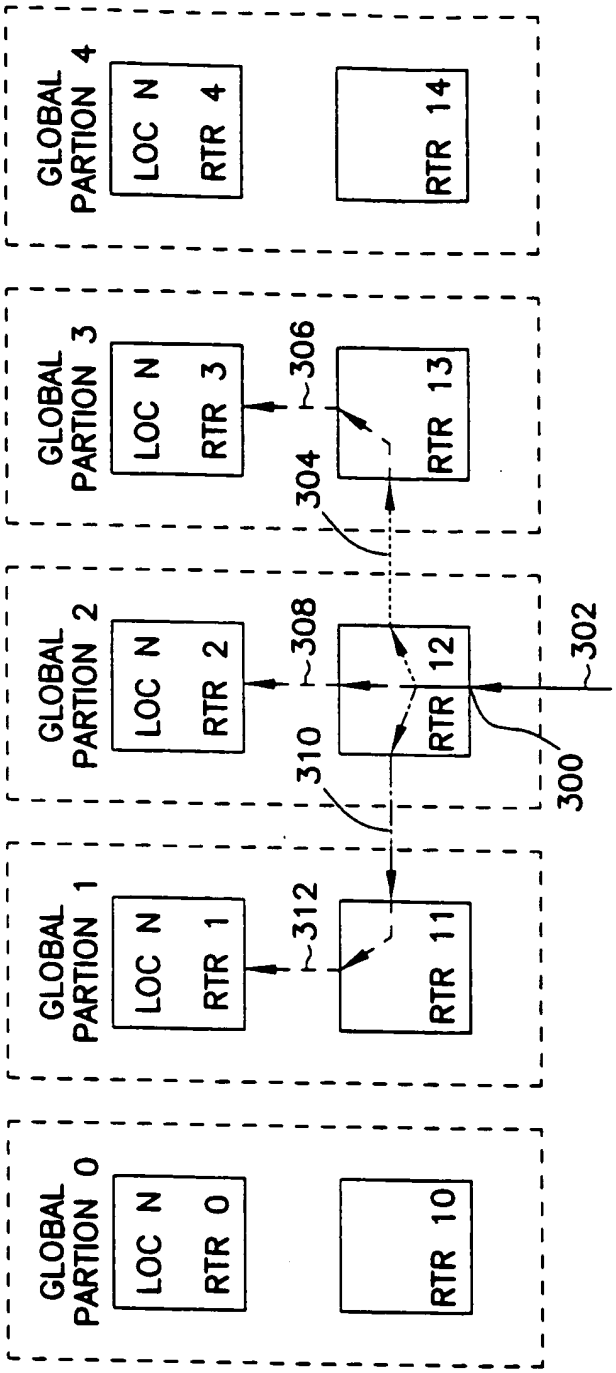

GLOBAL PARTION 4
LOC N RTR 4
RTR 14
GLOBAL PARTION 3
LOC N RTR 3
306
RTR 13
304
GLOBAL PARTION 2
LOC N RTR 2
308
RTR 12
302
310
300
GLOBAL PARTION 1
LOC N RTR 1
312
RTR 11
GLOBAL PARTION 0
LOC N RTR 0
RTR 10

FIG. 20

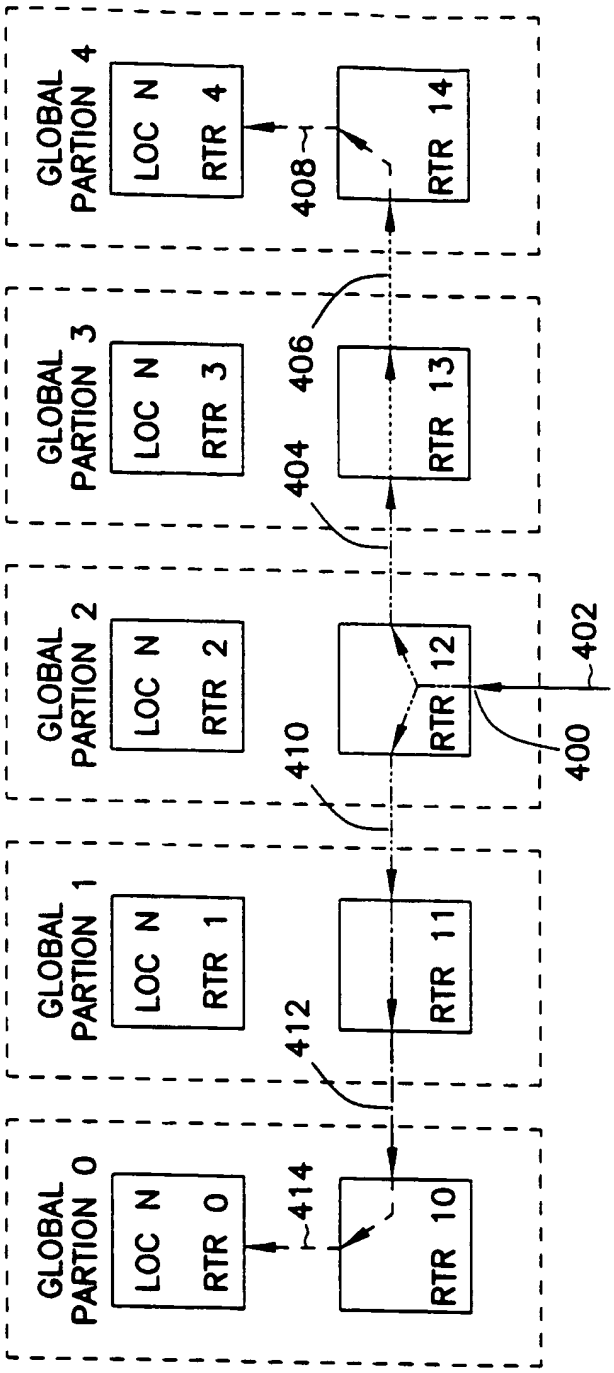

GLOBAL PARTION 4
LOC N RTR 4
RTR 14
408
GLOBAL PARTION 3
LOC N RTR 3
RTR 13
406
404
GLOBAL PARTION 2
LOC N RTR 2
RTR 12
402
400
410
GLOBAL PARTION 1
LOC N RTR 1
RTR 11
412
GLOBAL PARTION 0
LOC N RTR 0
RTR 10
414

FIG. 21

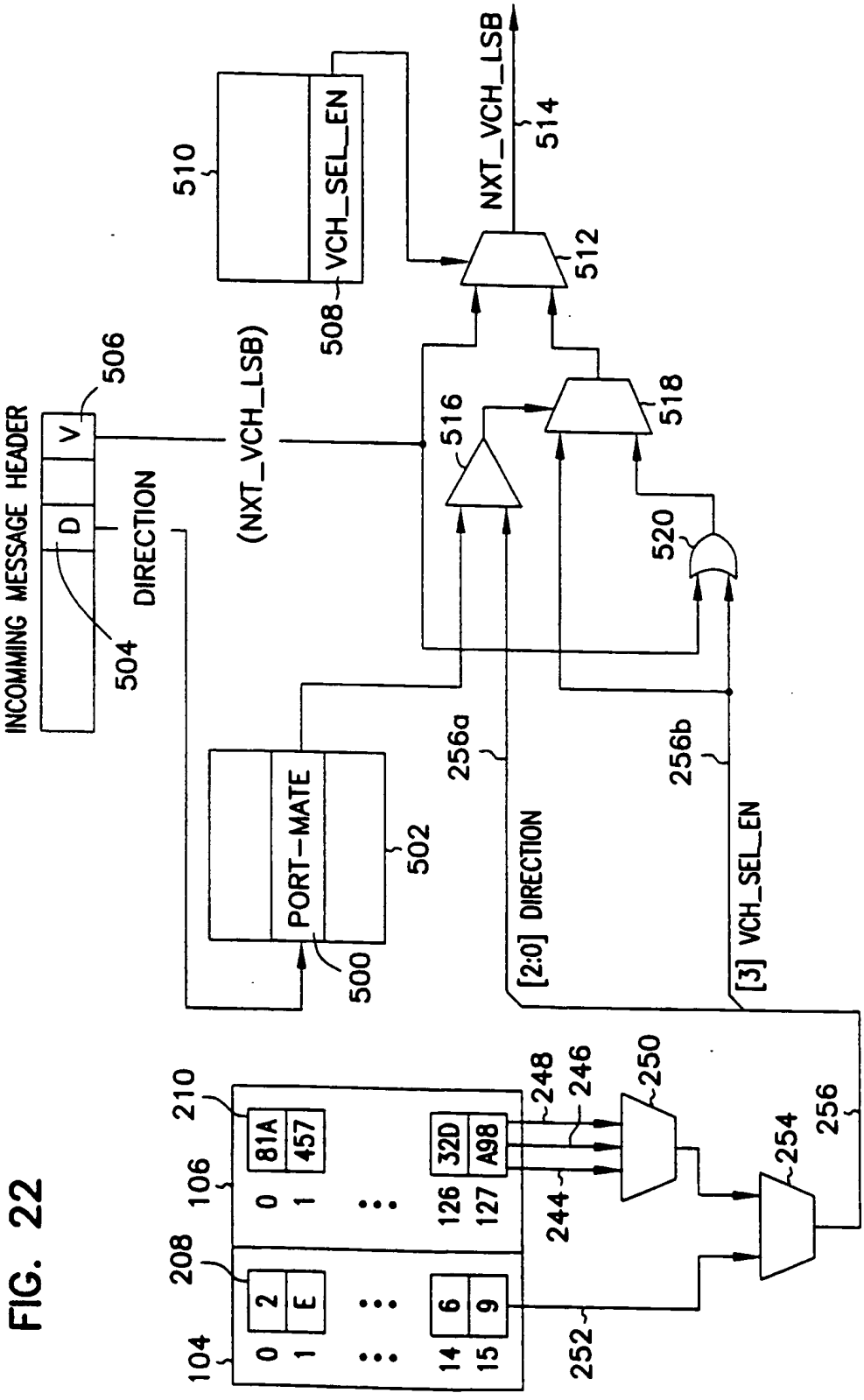

INCOMMING MESSAGE HEADER
V
D
506
504
DIRECTION
(NXT_VCH_LSB)
510
VCH_SEL_EN
508
NXT_VCH_LSB
514
512
516
518
520
PORT-MATE
502
500
256a
256b
[2:0] DIRECTION
[3] VCH_SEL_EN
210
106
208
104
0
1
14
15
126
127
81A
457
32D
A98
2
E
6
9
248
246
250
244
252
254
256

FIG. 22

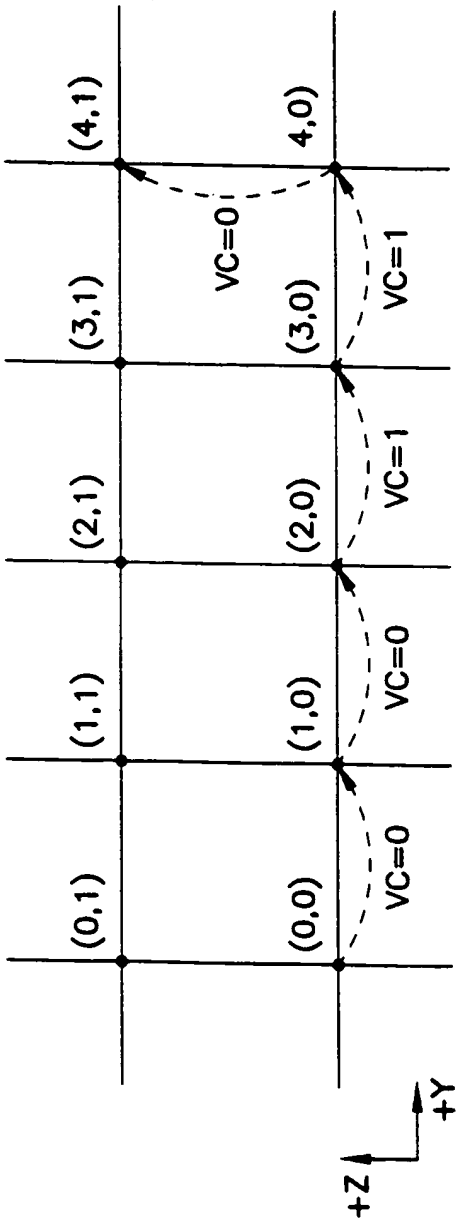


FIG. 23

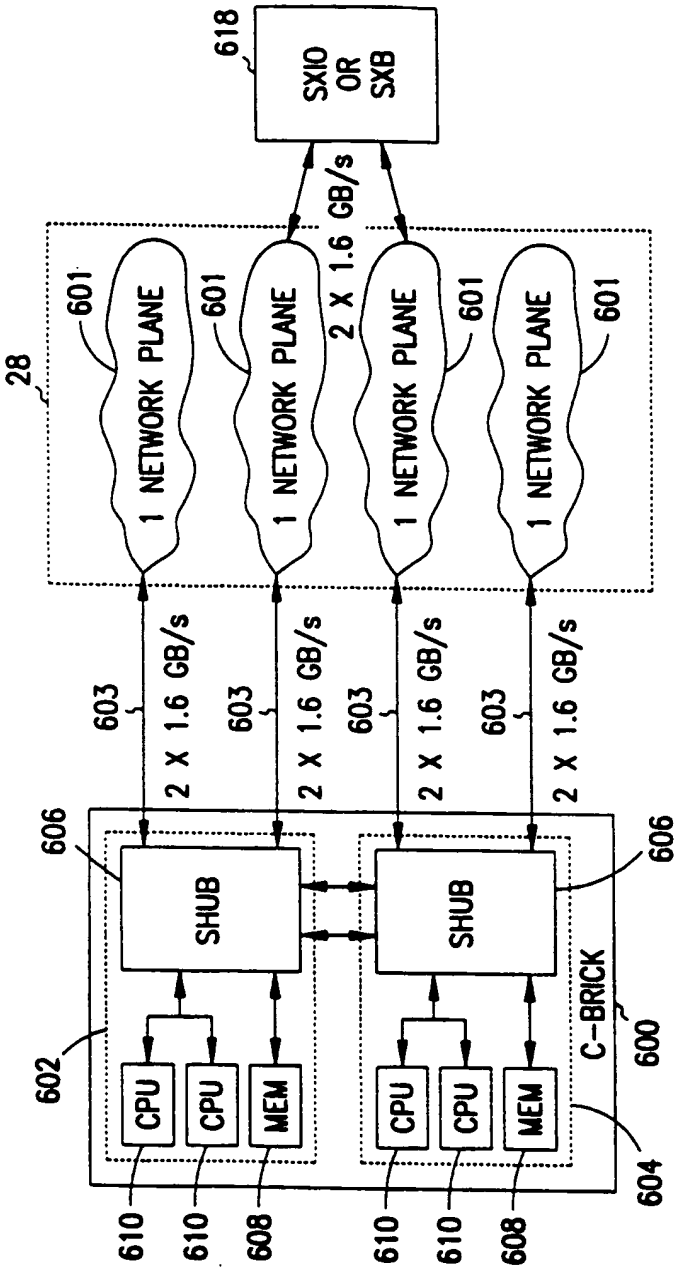

618
SXIO OR SXB
2 X 1.6 GB/s
28
601
1 NETWORK PLANE
603
606
SHUB
602
CPU
MEM
C-BRICK
600
604
610
608

FIG. 24

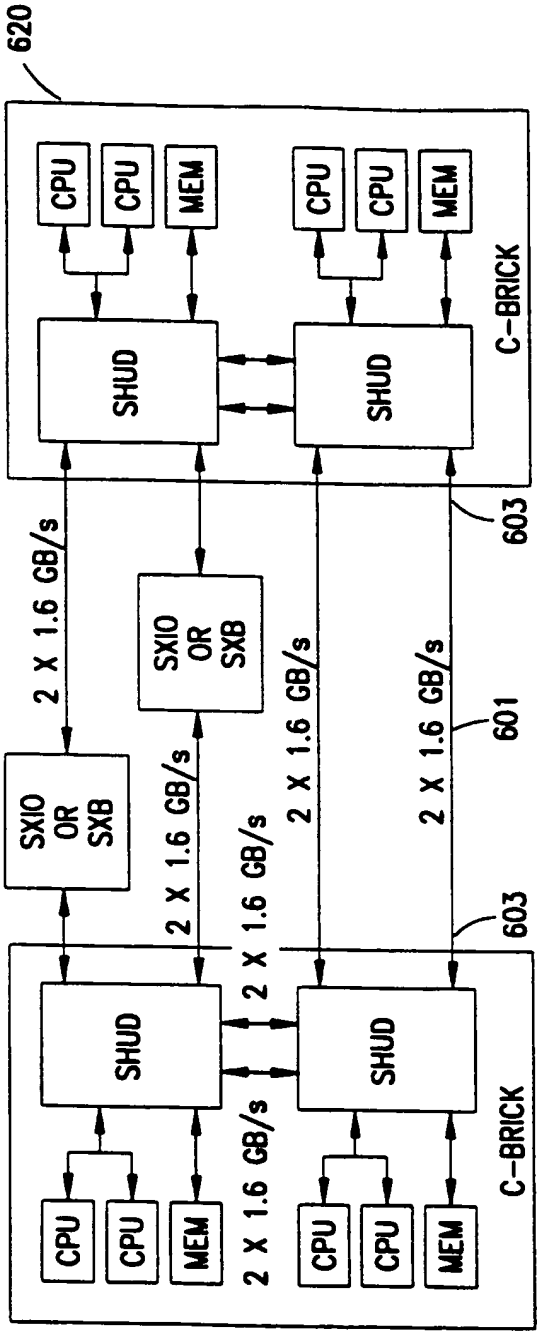

620
CPU
CPU
MEM
CPU
CPU
MEM
SHUD
SHUD
C-BRICK
603
2 X 1.6 GB/s
SXIO OR SXB
2 X 1.6 GB/s
2 X 1.6 GB/s
2 X 1.6 GB/s
601
SXIO OR SXB
2 X 1.6 GB/s
603
SHUD
SHUD
2 X 1.6 GB/s
C-BRICK
CPU
CPU
MEM
CPU
CPU
MEM

FIG. 25

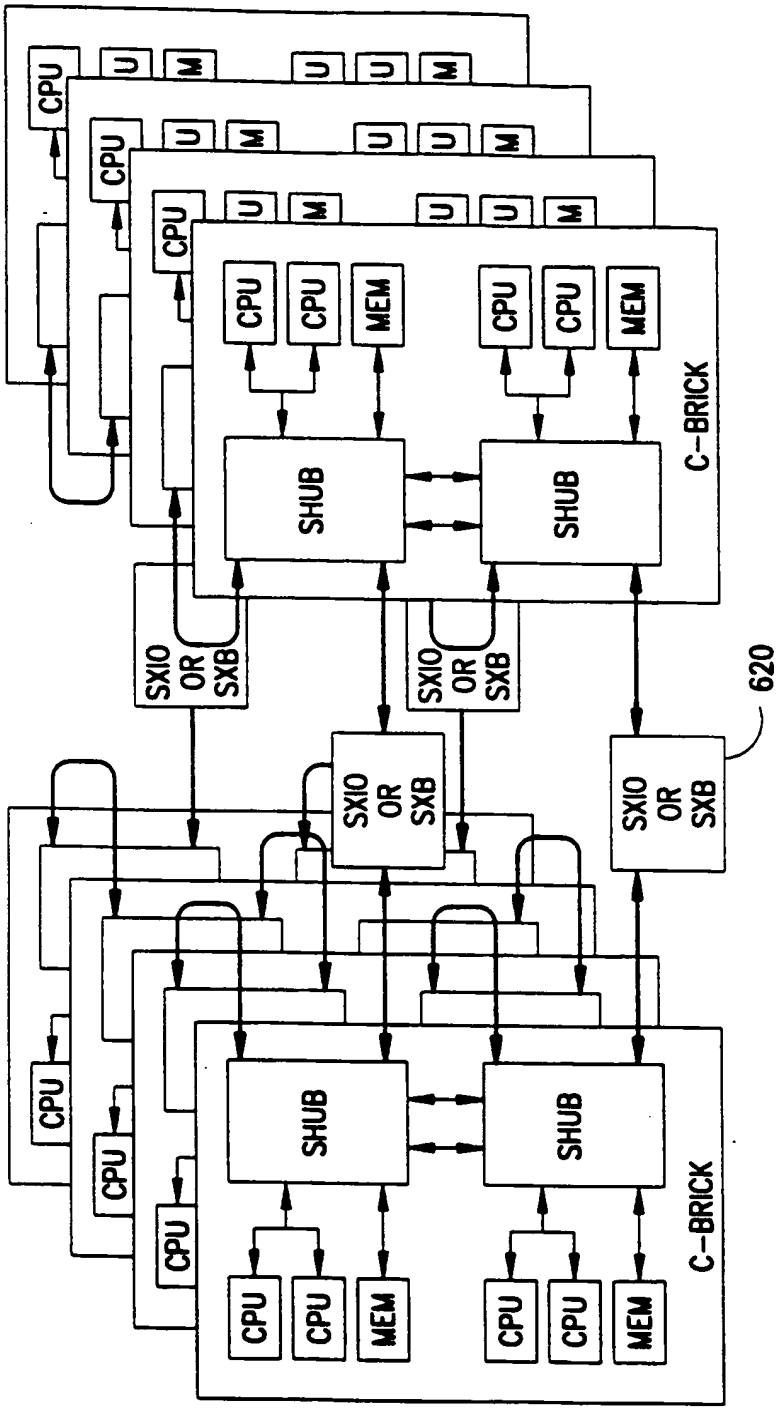


FIG. 26

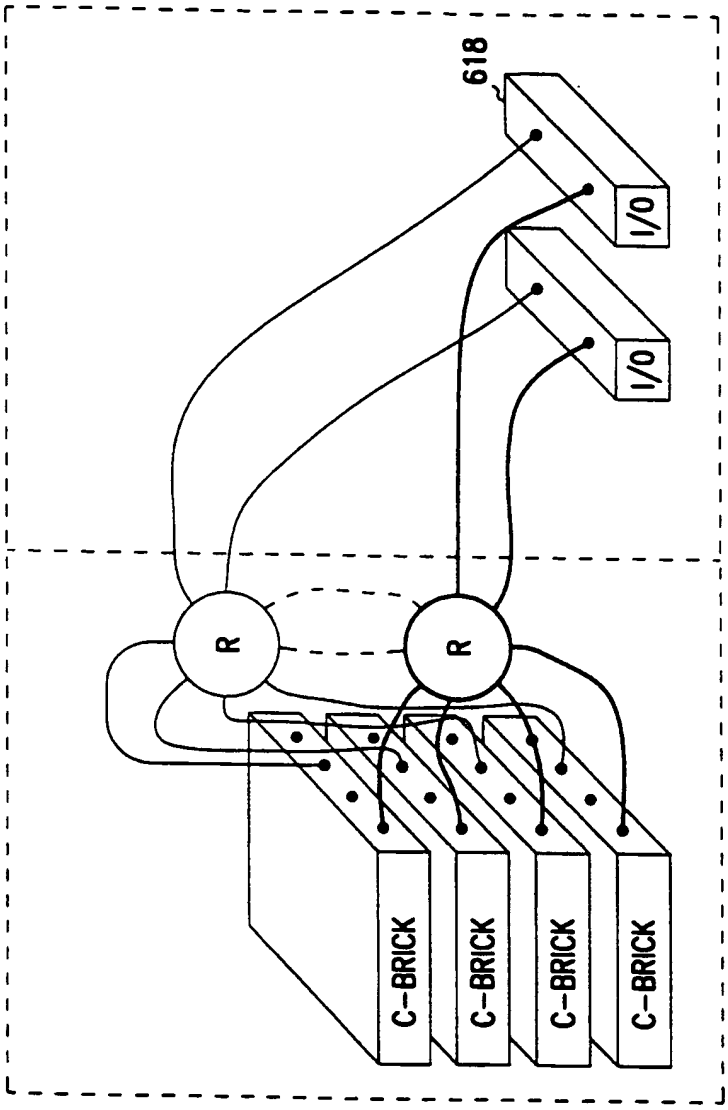


FIG. 27

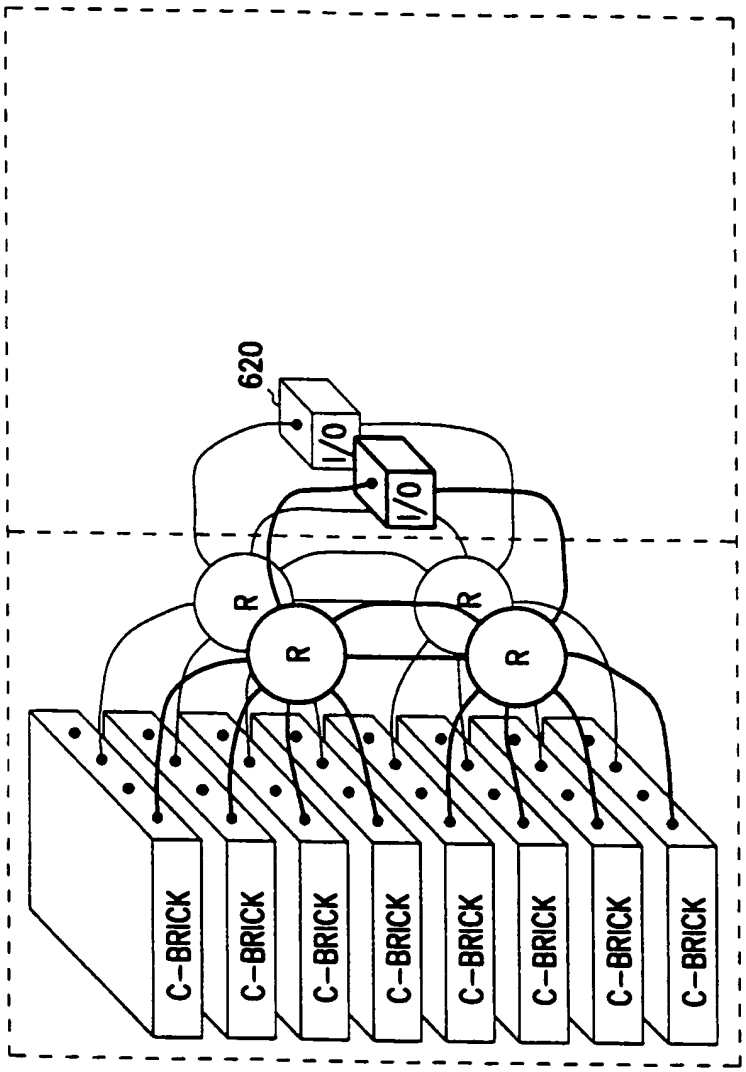

620
I/O
I/O
R
R
R
R
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK

FIG. 28

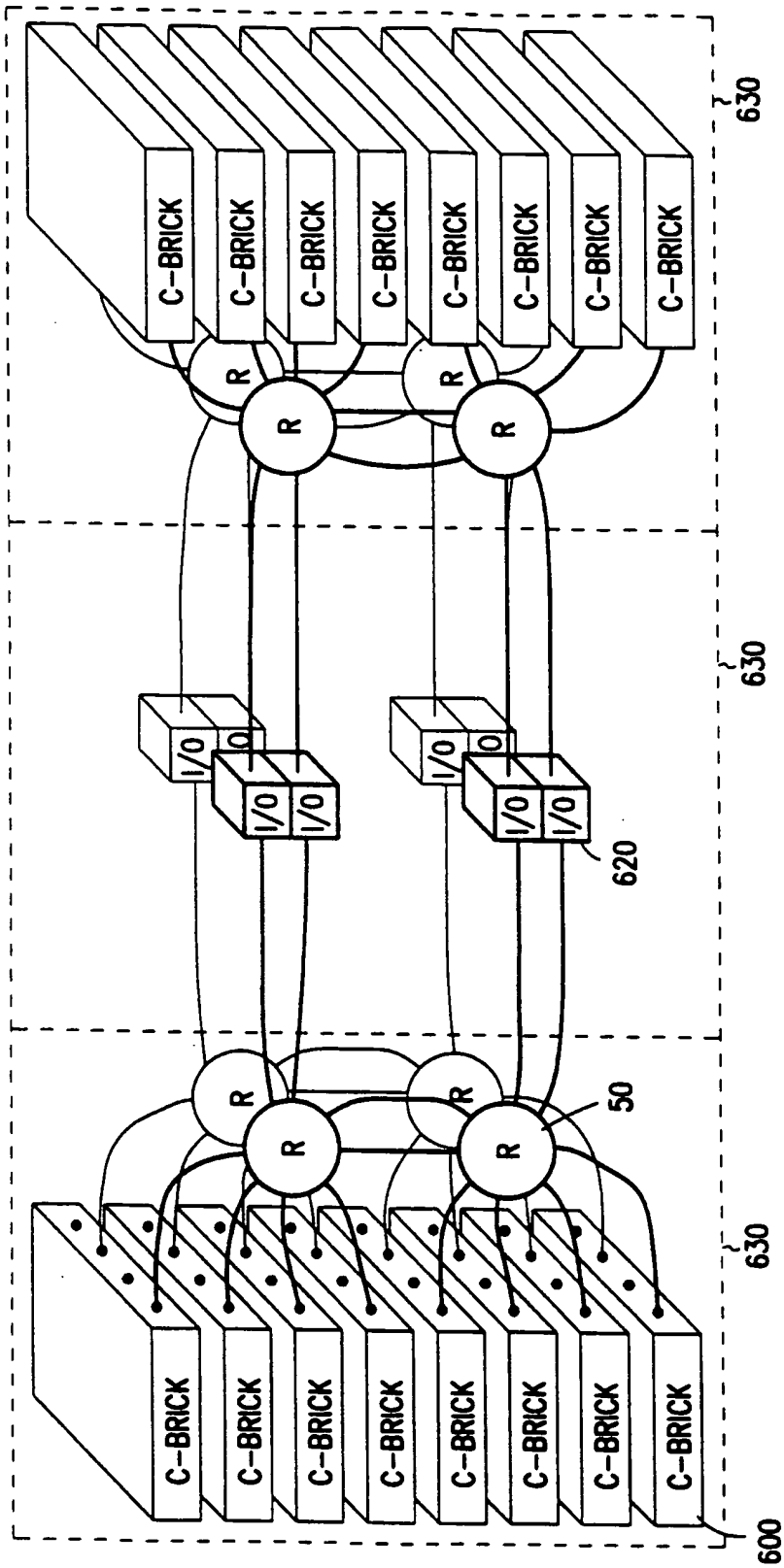


FIG. 29

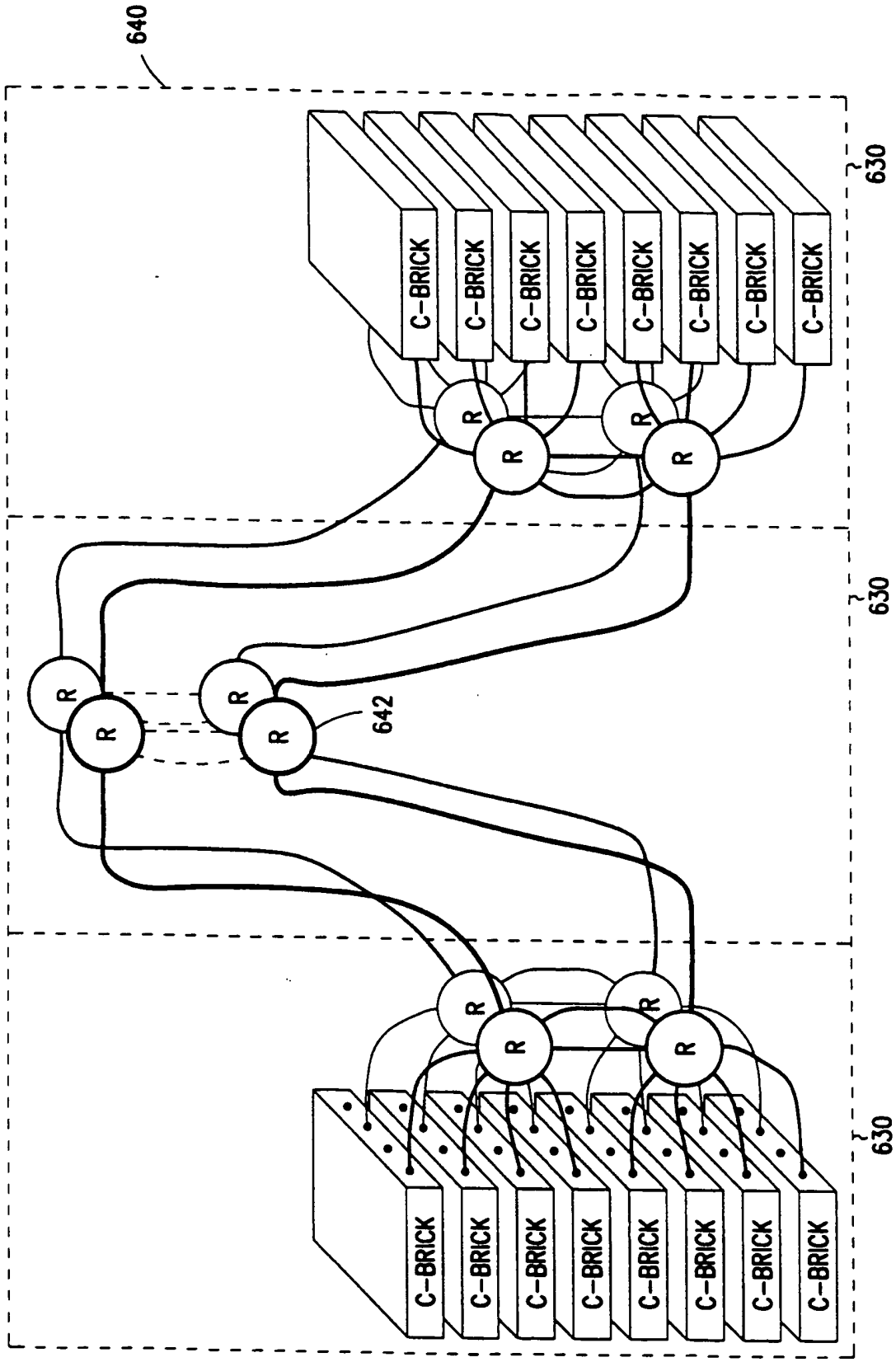

640
630
630
630
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
R
642

FIG. 30

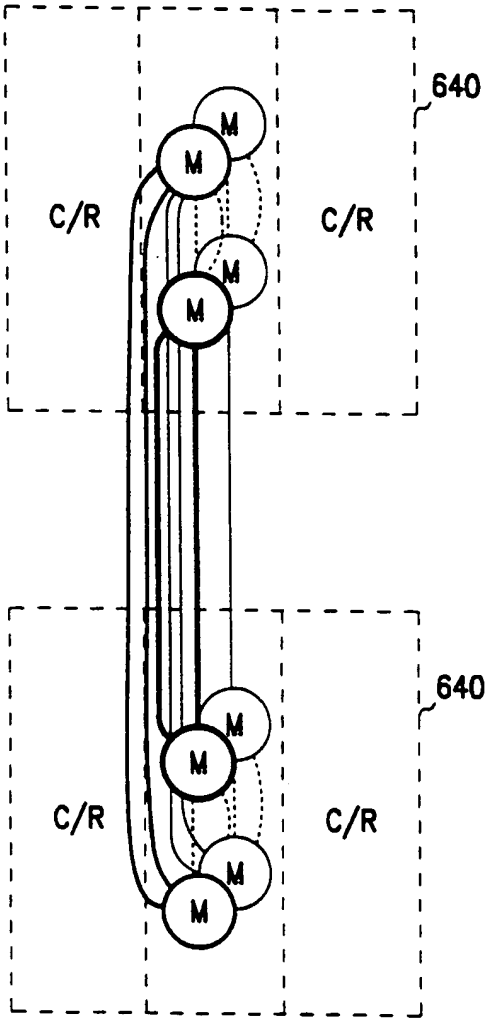


FIG. 31

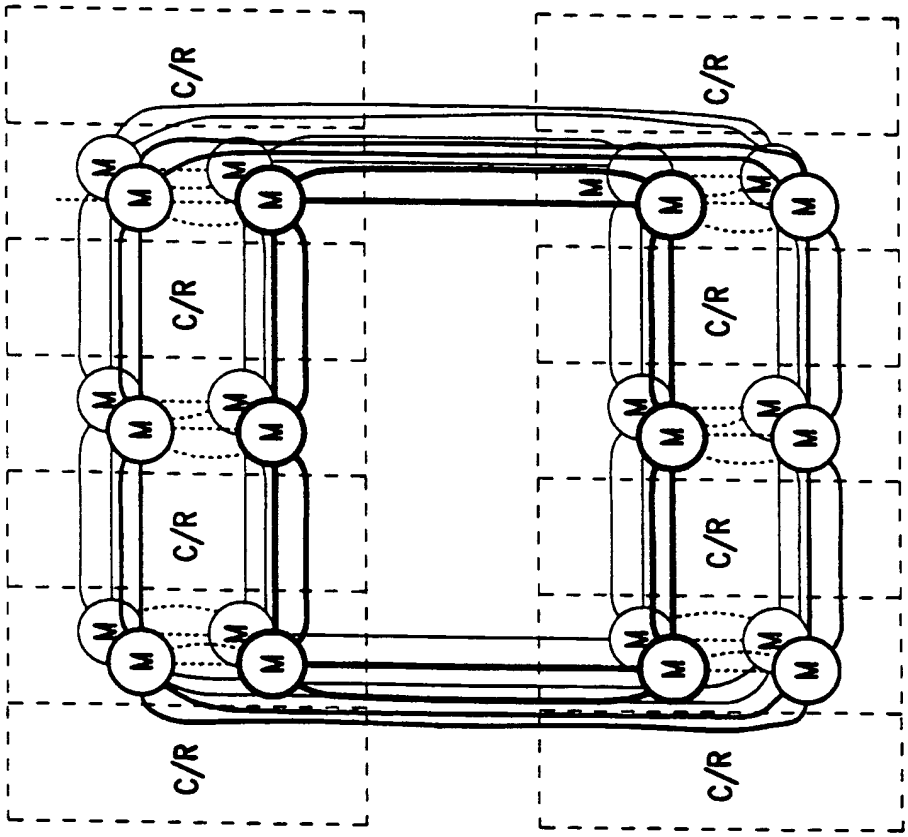


FIG. 32

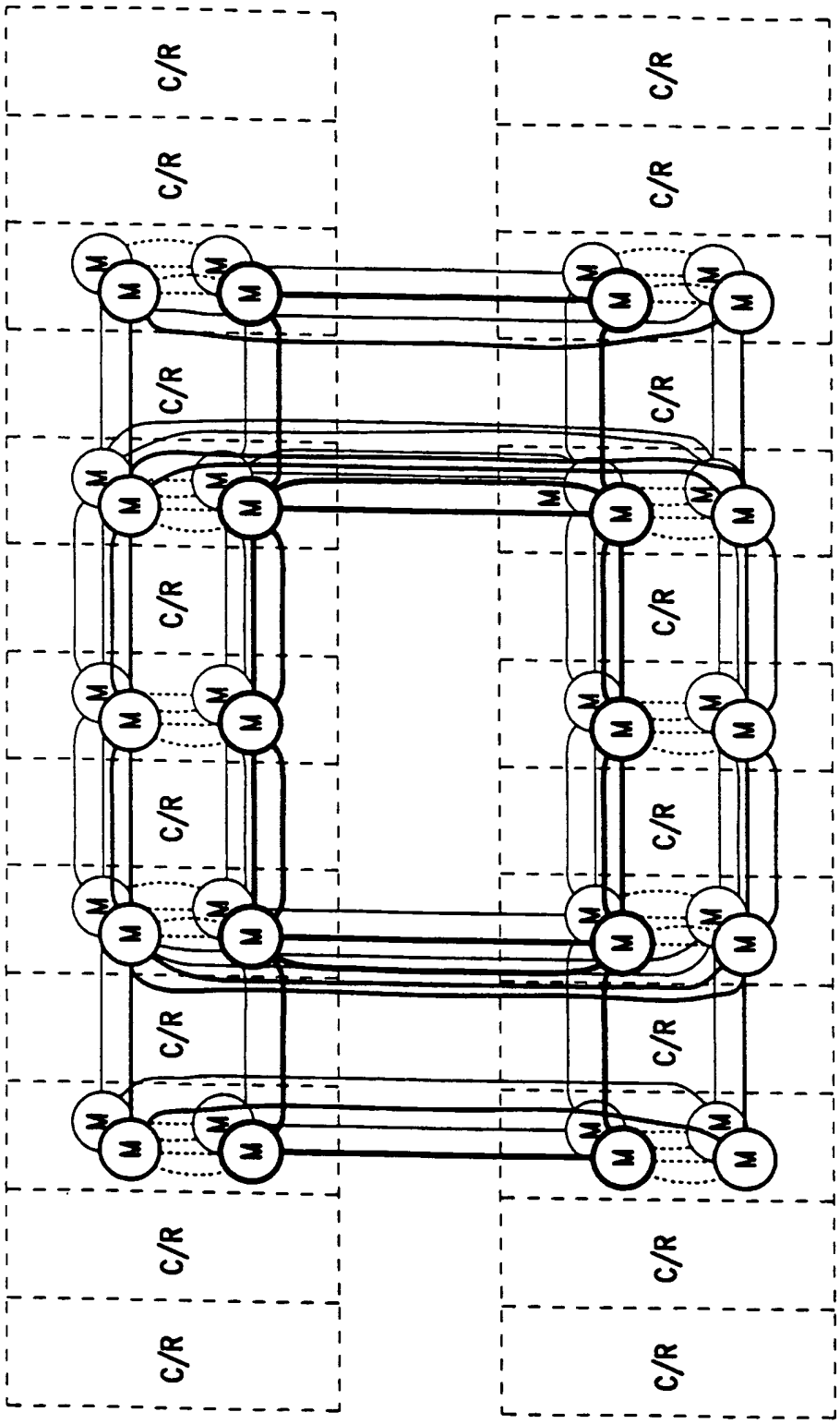


FIG. 33

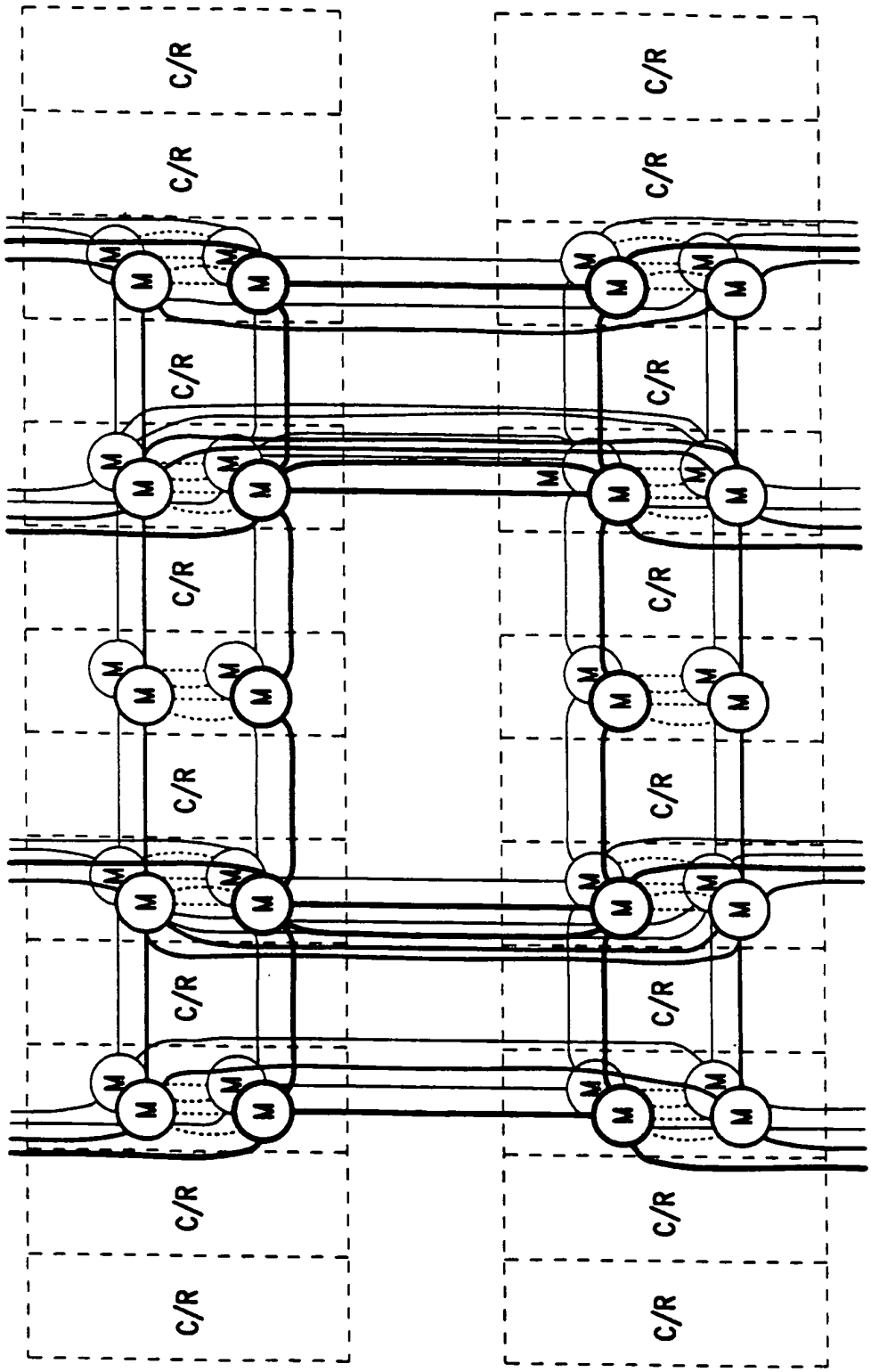


FIG. 34

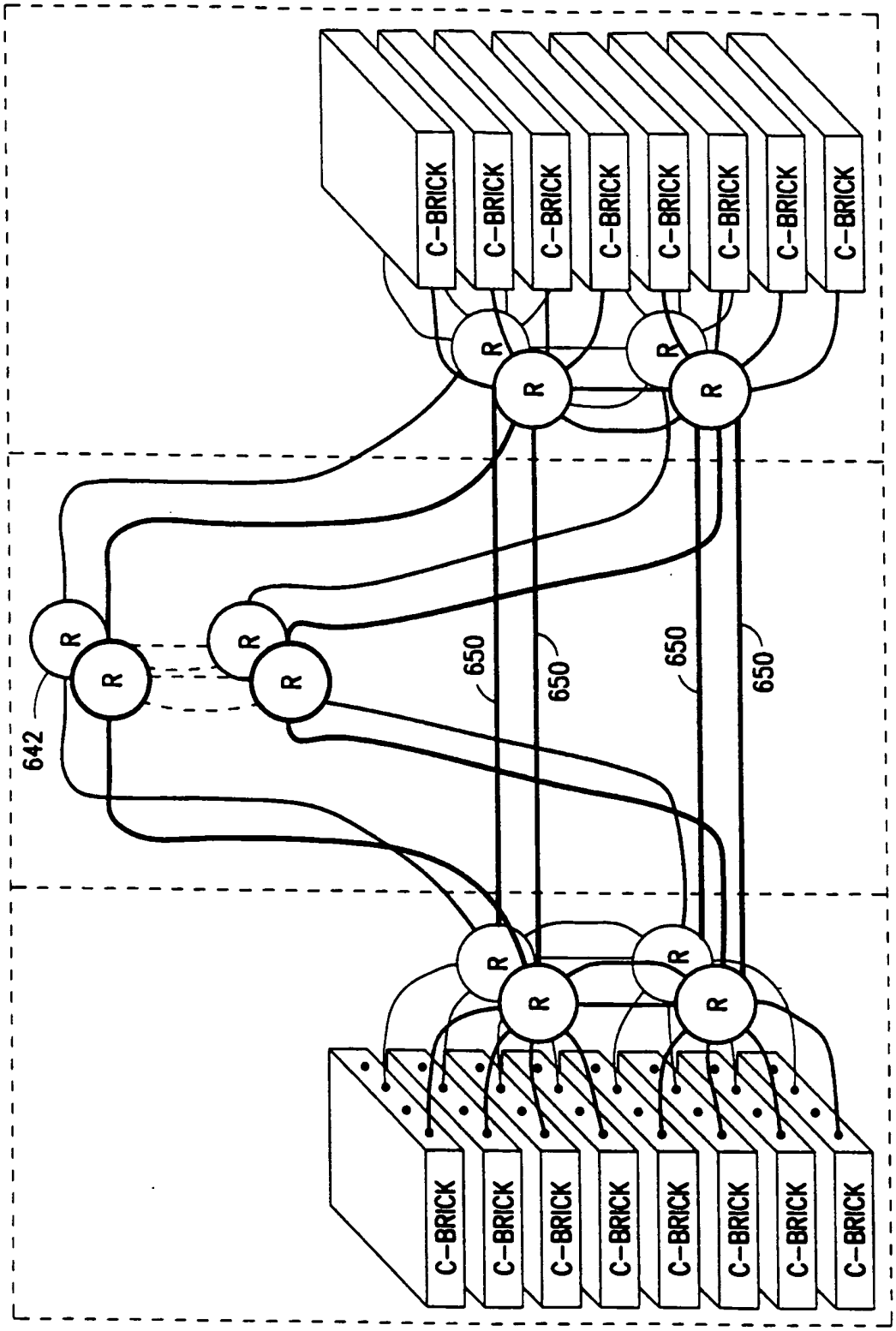

C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
R
642
650
650
650
650
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK

FIG. 35

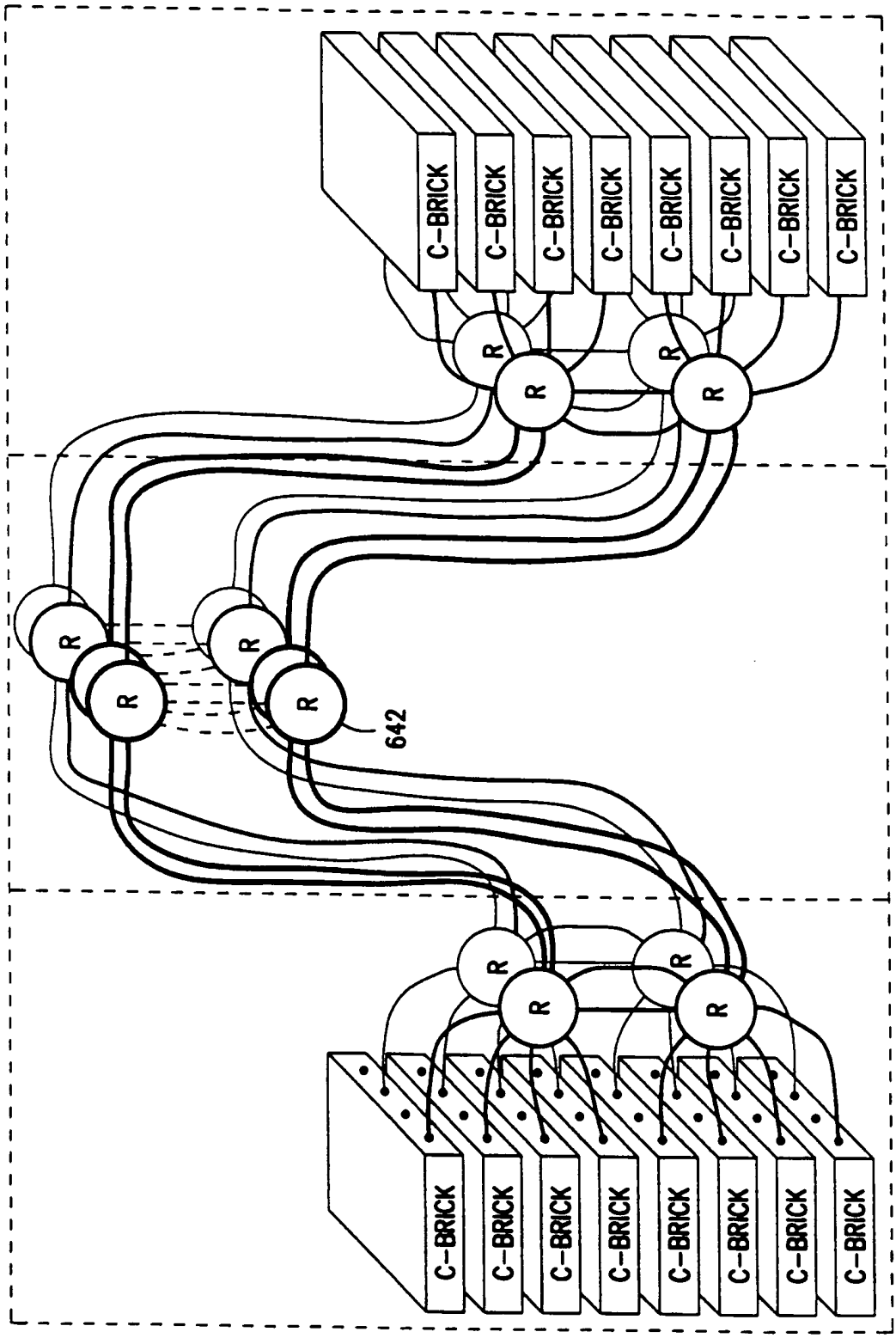

C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
R
642
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK

FIG. 36

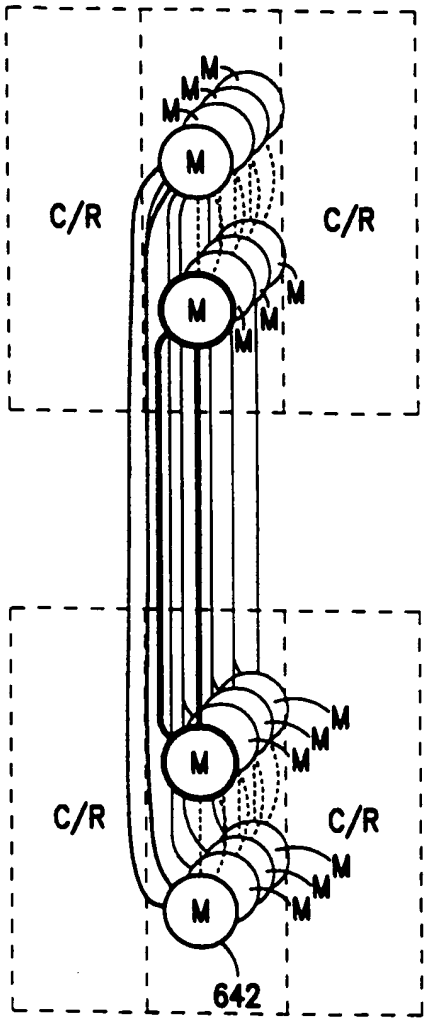


FIG. 37

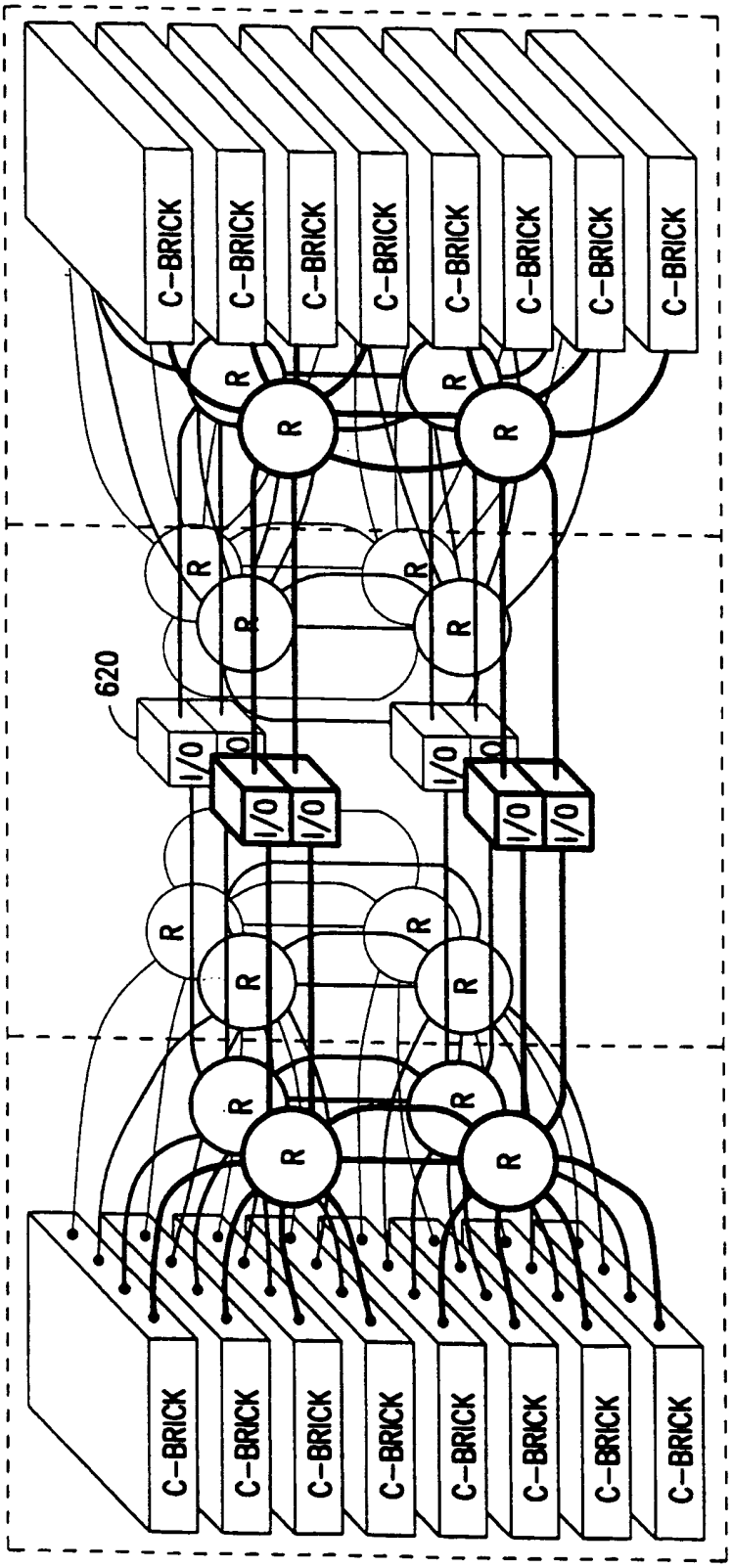

C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
R
620
I/O
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK

FIG. 38

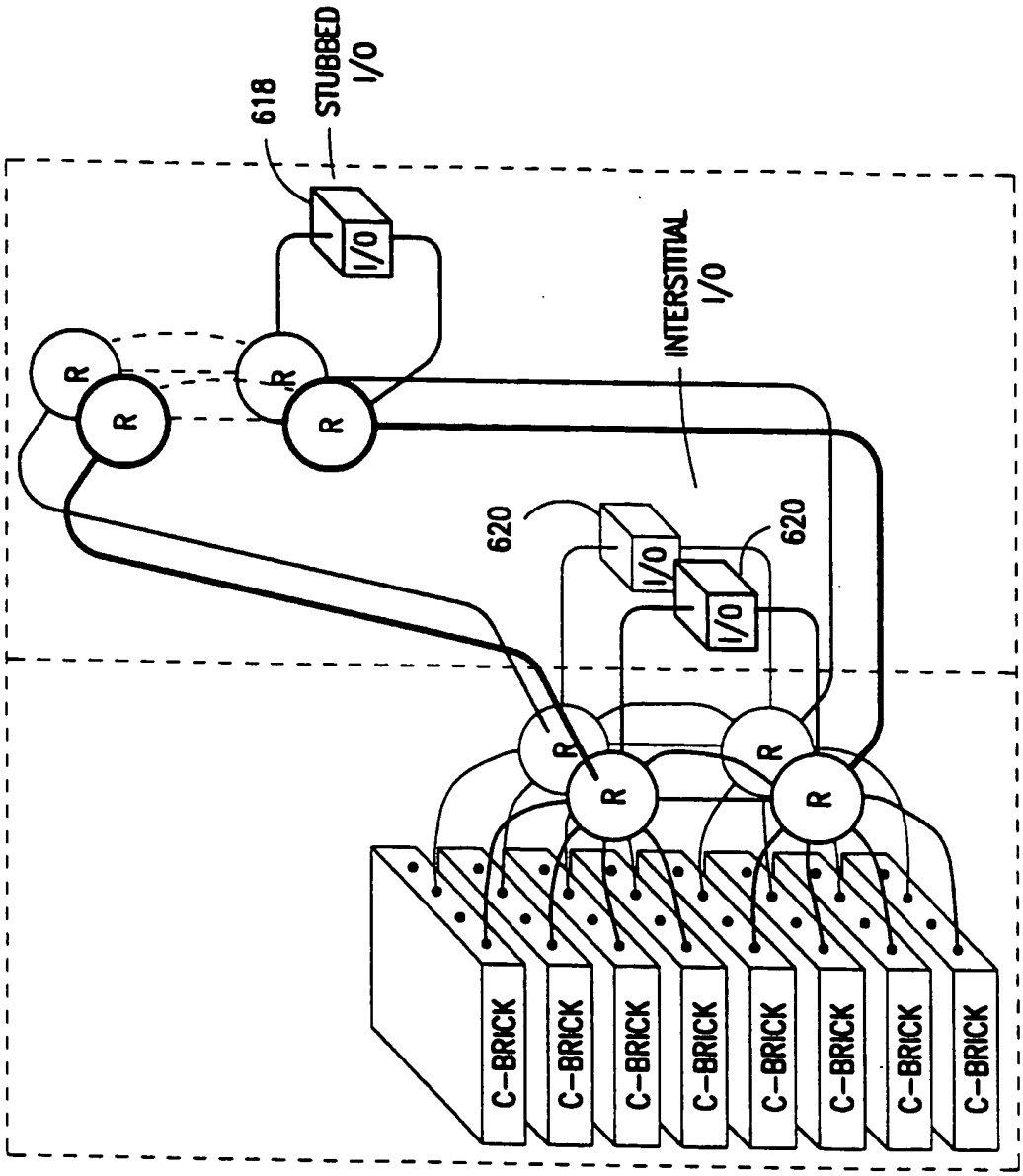

618
STUBBED I/O
I/O
INTERSTITIAL I/O
620
620
I/O
I/O
R
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK
C-BRICK

FIG. 39